

Learning Summary Report

Name: Ziyue Meng
Student ID: 36035432
Date: 24/10/2025

Self-Assessment

Grade demonstrated: **Distinction**

Declaration

I declare that this portfolio is my individual work. I have not copied from any other student's work or from any other source except where due acknowledgment is made explicitly in the text, nor has any part of this submission been written for me by another person. The extent to which I have used generative AI is described and appropriately attributed.

Signature: Ziyue Meng

My Journey

My Programming Journey: From Foundations to Mastery

Reflecting on my programming journey feels like tracing a path of continuous challenge and growth. My starting point was a blank slate, filled with curiosity but little practical experience. This journey began with C#, which served as my introduction to the structured world of programming. Learning about object-oriented principles, syntax, and basic algorithms in C# provided me with a solid, managed foundation. It was a challenging but manageable entry point into the world of coding.

The real test began when I transitioned to C++. Moving from the relative safety of C# to the raw power and complexity of C++ was a significant leap. Concepts like manual memory management, pointers, and more

intricate syntax rules presented a steep learning curve. The most formidable challenge I faced was undoubtedly the C3 assignment. I struggled with it for what felt like an eternity, spending countless hours debugging and refining my code. I would write a section, only to be met with a cascade of errors or logical flaws. There were numerous late nights spent reworking algorithms, tracing through code line by line, and questioning my approach. The process of repeatedly modifying my code to finally make it run correctly was exhausting, but the moment it finally executed flawlessly was incredibly rewarding. It taught me resilience and the critical importance of problem-solving and debugging over merely writing code.

My academic progress also faced a significant hurdle with the second exam, which focused exclusively on structures and arrays in C++. My first attempt was unsuccessful, and it served as a stark revelation. It forced me to confront the reality that my understanding of these fundamental data structures was superficial. I could define them in theory, but I struggled to apply them effectively to solve practical problems. I hadn't yet mastered how to manipulate arrays efficiently or how to design and utilize `structs` to organize data logically within a program.

Fortunately, during these challenging times, I was supported by an incredible teacher and a group of supportive classmates. My teacher was always patient, offering clarity and guidance that helped demystify complex topics. My peers were equally invaluable; we formed study groups where we could brainstorm solutions to problems like the C3

assignment and share insights for the exam. Their help was instrumental after I failed my first attempt. I revisited the concepts of structures and arrays with a new perspective, practiced extensively, and ultimately passed the exam on my second try. I am profoundly grateful for their support, which taught me the immense value of collaboration and seeking help when needed.

Looking back, I see far more than just a sequence of languages learned—C# laying the foundation, C++ teaching me depth and rigor, and Python yet to be explored. I see a transformation in my problem-solving mindset. I have evolved from a beginner who simply followed syntax to a budding developer who can think critically, endure through difficulties, and independently deconstruct and solve problems. This journey, paved with both struggles and breakthroughs, has not only equipped me with technical skills but also with the perseverance and humility necessary for continuous learning in the field of computer science.

Statement of Achievements

Achieved Grade Persona

I have successfully achieved the main objectives outlined in my P1 learning plan. While I adjusted my target from HD to Distinction, I developed strong programming skills in C#, C++, and Python through consistent practice and assignment completion. When facing debugging challenges, I occasionally used AI tools to identify errors after exhaustive personal attempts, which significantly improved my problem-solving approach.

I have met the Distinction standards through demonstrated conceptual understanding and skill application. My honest self-assessment recognizes areas for growth while acknowledging solid achievement in core programming concepts. The progress shown in assignments and exams, particularly with C++ structures and arrays, reflects the consistent development expected at this level. This grade represents both my current capabilities and a foundation for future improvement.

Achievement of Unit Learning Outcomes

Throughout the P1-P11 curriculum, I have thoroughly mastered the core syntax and concepts of all three programming languages, including structures, classes, and enums. I accomplished this through completely independent study, without relying on external assistance.

In the C1 and C2 modules, I delved into more advanced topics such as pointers and memory allocation, gaining a deep understanding of how code interacts with computer systems. I not only grasped these concepts fully but can also confidently apply them in personal projects.

The C3 assignment introduced me to code testing methodologies. While I could write the initial code structure, I encountered significant challenges during execution, including runtime errors and infinite loops. To overcome these, I sought guidance from my instructor and utilized Gen AI to help identify specific errors. After considerable effort and iterative debugging, I successfully resolved all issues. This process, though demanding, proved immensely educational.

For the C4 task involving CSV file processing, my background in Applied Data Science allowed me to connect the required syntax with my prior knowledge from professional courses. This familiarity enabled me to complete the assignment efficiently.

The D1-D4 independent project presented substantial challenges. Over approximately three weeks, I continuously refined my code to eliminate errors. While my expertise in pandas, numpy, and matplotlib facilitated data-related components, I utilized Gen AI to assist with TKinter implementation, simultaneously studying its syntax throughout the process. This experience has prepared me to handle similar tasks with greater confidence in the future.

Reflections

Most important thing learnt

The most important thing I learned was a systematic approach to debugging and problem-solving. During the C3 assignment, although I could write the code structure, I encountered numerous errors and infinite loops during debugging. By consulting my teacher and using Gen AI to assist with analysis, I learned how to methodically locate problems, analyze the root causes of errors, and implement effective solutions. This skill is far more important than merely mastering syntax; it gives me the confidence to face any programming challenge.

Most interesting topic or task

The content on pointers and memory management in C1 and C2 was the most fascinating to me. This knowledge allowed me to truly understand, for the first time, how code interacts with the computer system. Through practical application, I not only grasped these concepts but could also apply them in my own projects. This leap from theory to practice gave me a great sense of accomplishment.

Challenges Overcome

The debugging process for the C3 assignment was the biggest challenge I faced. When the code repeatedly produced errors, I adopted a step-by-step debugging strategy: first independent analysis, then seeking guidance from the teacher, and finally using Gen AI to help pinpoint specific issues. After multiple rounds of iterative modifications, I finally succeeded in resolving the problems. This process taught me the importance of patience and systematic thinking.

What next?

I am eager to learn more about Graphical User Interface (GUI) development and related frameworks. Although I used Tkinter in this unit, I realize this is just the beginning. The programming foundation laid in this unit has equipped me with the ability to further explore more complex GUI frameworks, which will help me develop real-world applications.

My initial learning plan

Looking back at the learning plan outlined in P1, I found it provided a clear direction for my studies, but required flexibility in its implementation during practice. Although I adjusted my final grade target from High Distinction to Distinction, I did achieve the core goal of mastering multiple programming languages, showing significant improvement, especially in practical application skills.

Retrospective Advice for Myself

If I could go back to the start of the unit, I would advise myself to start practical projects earlier and establish a systematic habit of logging errors. Furthermore, I

would encourage myself not to be afraid to seek help earlier, as collaborative learning often leads to breakthrough progress.

If I could change one thing ...

If I could change one thing about this unit, I would wish for more dedicated guidance on debugging techniques and best practices. Although I learned a great deal through hands-on experience, more systematic instruction on debugging methodologies might have helped me overcome the challenges in the C3 assignment more quickly.

Other

I would like to express my sincere gratitude to all the doctors—both my instructor and the doctors on Ed who provided timely solutions—for their invaluable support throughout this semester. Your guidance and patience were instrumental in helping me thoroughly grasp all the key concepts of this unit. Thanks to your assistance, I was able to overcome numerous challenges and deepen my understanding of programming. This semester has been a truly rewarding journey, and I have grown significantly both academically and personally. Thank you for making this learning experience so meaningful and impactful.

MONASH

INTRODUCTION TO PROGRAMMING

ZIYUE MENG

Portfolio Submission

Submitted By:
Ziyue MENG
zmen0034

Tutor:
Andrew CAIN

October 24, 2025



Contents

1 Learning Summary Report	1
2 Overall Task Status	2
3 Learning Outcomes	3
3.1 ULO1 Use Instructions	3
3.2 ULO2 Programs, Functions, and Types	4
3.3 ULO3 Use libraries	5
3.4 ULO4 Use good programming practices	6
4 Learning Plan	7
5 Building and Running	13
6 Sequence and Data	26
7 Control Flow	43
8 C/C++	67
9 Structuring Data	84
10 Test 1	103
11 Indirect Access	104
12 Structuring Code	131
13 Working with Multiples	155
14 Memory Management	196
15 Custom Program - Pitch	259
16 Programming Project	267
17 More Python	300
18 Active Engagement	332
19 Custom Program - Progress	346
20 Custom Program - Final	377
21 Something Awesome - Pitch	423
22 Custom Program - HD Report	424
23 Classes and Objects	425
24 Inheritance and Polymorphism	452
25 Learning Summary	474
26 Another Language - Python	483
27 Test 2	504

2 Overall Task Status

Task	Status	Times Assessed
Learning Plan	Complete	1
Building and Running	Complete	3
Active Engagement	Discuss	1
Sequence and Data	Complete	1
Control Flow	Complete	2
Test 1	Complete	1
C/C++	Complete	2
Structuring Code	Complete	4
Structuring Data	Complete	3
Working with Multiples	Complete	7
Indirect Access	Complete	4
Memory Management	Complete	5
Custom Program - Pitch	Complete	1
Test 2	Complete	1
Programming Project	Complete	1
Another Language - Python	Complete	1
Custom Program - Progress	Complete	1
Classes and Objects	Complete	2
More Python	Complete	1
Something Awesome - Pitch	Not Started	0
Inheritance and Polymorphism	Complete	1
Custom Program - Final	Assess in Portfolio	0
Custom Program - HD Report	Not Started	0
Something Awesome - Final	Not Started	
HD Project Showcase	Not Started	
Learning Summary	Assess in Portfolio	0

3 Learning Outcomes

3.1 ULO1 Use Instructions

Effectively use variables, conditionals and loops in computer programs

Task	Status	Times Assessed
Sequence and Data	Complete	1
Control Flow	Complete	2
C/C++	Complete	2
Indirect Access	Complete	4
Custom Program - Pitch	Complete	1

3.2 ULO2 Programs, Functions, and Types

Design, construct and test simple programs that include user-defined functions and data structures

Task	Status	Times Assessed
C/C++	Complete	2
Structuring Code	Complete	4
Structuring Data	Complete	3
Indirect Access	Complete	4
Custom Program - Pitch	Complete	1

3.3 ULO3 Use libraries

Write programs efficiently by discovering and leveraging libraries

Task	Status	Times Assessed
C/C++	Complete	2
Structuring Data	Complete	3
Indirect Access	Complete	4
Custom Program - Pitch	Complete	1

3.4 ULO4 Use good programming practices

Implement good programming practices including unit testing, basic documentation and readability.

Task	Status	Times Assessed
Learning Plan	Complete	1
Building and Running	Complete	3
Sequence and Data	Complete	1
Control Flow	Complete	2
C/C++	Complete	2
Structuring Code	Complete	4
Structuring Data	Complete	3
Indirect Access	Complete	4
Custom Program - Pitch	Complete	1

4 Learning Plan

This task helps you connect with FIT1045 Introduction to Programming, its assessment, and the planning necessary to achieve different grade outcomes.

Outcome	Description		
Date	Author	Comment	
TLO1		Reflect on prior programming knowledge and articulate goals	
TLO2		Develop a personal study plan and justify how it aligns with your learning goals	
2025/07/27 19:11	Ziyue Meng	Working On It	
2025/07/27 19:11	Ziyue Meng	Not Started	
2025/07/28 11:08	Ziyue Meng	Working On It	
2025/07/28 11:38	Ziyue Meng	Ready for Feedback	
2025/07/31 17:53	Ganesh Krishnasamy	Fix and Resubmit	
2025/07/31 17:54	Ganesh Krishnasamy	which programming language that u know.	
2025/07/31 17:55	Ganesh Krishnasamy	Complete	
2025/07/31 19:43	Ziyue Meng	In fact, I have only learned the Python language, and not in - depth. I have only studied the basic language and various loops, and have not learned other programming languages.	

INTRODUCTION TO PROGRAMMING

ONTRACK SUBMISSION

Learning Plan

Submitted By:

Ziyue MENG
zmen0034
2025/07/28 01:38

Tutor:

Ganesh KRISHNASAMY

Task Outcomes	Supports
TLO1 Reflect on prior programming knowledge and articulate goals for learning in this unit.	ULO4 Use good programming practices
TLO2 Develop a personal study plan and justify how it aligns with your target grade and unit structure, including engagement with feedback, submission timelines, and assessment requirements.	ULO4 Use good programming practices

- TLO1 Reflect on prior programming knowledge and articulate goals for learning in this unit.
- TLO2 Develop a personal study plan and justify how it aligns with your target grade and unit structure, including engagement with feedback, submission timelines, and assessment requirements.

July 28, 2025



Learning Plan

Name: Ziyue Meng
Student ID: 36035432
Date: 28/7/2025

Personal Learning Goals

My prior programming and computing experience

I have built a solid foundation in basic programming, being able to handle tasks like writing simple scripts for data processing. However, when it comes to other computer languages such as Java, C++, I am completely unfamiliar. I have no practical experience in writing code with them, nor do I understand their syntax rules, unique features, or application scenarios.

My goals for FIT1045, and how this will support my career goal

My primary goal in taking this course is to gain proficiency in more computer languages beyond Python. I hope to not only learn the syntax and basic usage of these languages but also delve into their underlying principles and best practices. Additionally, I aim to enhance my overall programming skills, including improving code efficiency, mastering debugging techniques, and learning how to design more logical and scalable programs. By achieving these, I want to be well-prepared for my future career, as having a diverse set of programming skills will undoubtedly make me more competitive in the job market, whether it's in software development, web programming, or other related fields.

My target grade and strategy to achieve it

I am determined to earn a High Distinction (HD) in this course. To reach this target, I will spare no effort in my studies. I will attend every class with full concentration, actively participate in discussions, take detailed notes to review later, and make sure I understand every concept and technique taught. When it comes to assignments, I will complete them independently, without any form of plagiarism or reliance on others. I will also take the initiative to do extra practice, explore additional learning resources, and seek help from the instructor or classmates promptly whenever I

encounter difficulties. I believe that through consistent hard work and dedication, I can achieve my academic goal.

My planned task submission dates

Task		Suggested Week Due by Grade				Latest
		Pass	Credit	Distinction	HD	
P1	Learning Plan	1	1	1	1	3
P2	Building and Running	1	1	1	1	3
P3	Sequence and Data	2	2	2	2	5
P4	Control Flow	4	4	3	3	6
T1	Test 1 (run in week 5)	5	5	5	5	6
P5	C / C++	5	5	4	4	6
P6	Structuring code	6	5	5	4	9
P7	Structuring data	7	6	5	5	9
C1	Indirect Access	-	7	6	5	10
P8	Working with multiples	8	8	7	6	10
C2	Deep Dive Memory	-	9	8	7	10
D2	Custom Program - Pitch	-	-	8	7	10
C3	Programming Project	-	10	9	8	11
T2	Test 2 (weeks 8, 10, and 12)	10	10	8	8	12
P9	Another Language - Python	10	10	9	8	12
P10	Classes and Objects	11^	11^	10	9	12
D3	Custom Program - Progress	-	-	10	9	12
P11	Inheritance and Polymorphism	12^	12^	11	10	12
C4	More Python	-	12^	11	10	12
H1	Something Awesome - Pitch	-	-	-	11	12
D4	Custom Program - Final	-	-	12^	11^	12
D1	Active Engagement	-	-	12^	11^	12
H2	Custom Program Report	-	-	-	11^	12
H3	Something Awesome - Final	-	-	-	12^	12
P12	Learning Summary	12^	12^	12^	12^	12
H4	High Distinction Showcase	-	-	-	14 	12

Week	Starts
1	28 Jul
2	4 Aug
3	11 Aug
4	18 Aug
5	25 Aug
6	1 Sept
7	8 Sept

Week	Starts
8	15 Sept
9	22 Sept (29 Sept break)
10	6 Oct
11	13 Oct
12	20 Oct
13	27 Oct
14	3 Nov

I understand how this unit works.

I, Ziyue Meng(36035432), confirm that I have read and understood how this unit works: I understand that:

- I will receive a grade in accordance with the tasks I complete.
- I can change my target grade at any time during the semester, up until I submit my portfolio.
- I am responsible for providing evidence of my work to satisfy the requirements in each module and task completed.
- I acknowledge that I need to sit and pass Test 1 and Test 2, and that I need to attend the test on campus at the indicated dates.
- I may be requested to respond to feedback and resubmit tasks to achieve the required standard.
- Following my initial submission, I will work with my tutor and the teaching team to update and complete my submissions until they demonstrate I have achieved the learning objectives of the module/task.
- I understand I will only receive feedback on my task submissions up until the task's latest due date in OnTrack. After this point, I will need to ensure it is up to the required standard within my portfolio.
- I understand that I will need to discuss my work with the teaching team in the applied classes to validate and enhance my understanding of the concepts.
- I have reviewed the timeline in the Learning Plan task sheet, and I have listed the dates I aim to submit my work for feedback, and update my plan in OnTrack.
- I commit to submit the initial submission of my work before the deadlines listed in the table above. I acknowledge that these deadlines can be modified by me, in agreement with my tutor, and that I am responsible for requesting these modifications as soon as I become aware of the need to do so, and no later than the deadline to be modified.

Name:Ziyue Meng
Student ID:36035432
Date:28/7/2025
Signature:Ziyue Meng

5 Building and Running

Before you can get started learning to program you need to make sure that you have things set up and can use the terminal to build and run programs.

Outcome	Description		
Date	Author	Comment	
TLO1		Document the process of setting up the programming environment.	
TLO2		Summarise key compilation concepts and terminal commands.	
TLO3		Reflect on your learning process and describe strategies you will use to improve.	
2025/07/30 10:46	Ziyue Meng	Working On It	
2025/07/30 12:25	Ziyue Meng	Ready for Feedback	
2025/08/11 12:57	Ganesh Krishnasamy	you need to show the journey, e.g. installation process, how to create your project files, to build and run them.	
2025/08/11 12:57	Ganesh Krishnasamy	also you need to the screen capture for HelloGui and C++ projects	
2025/08/11 12:57	Ganesh Krishnasamy	Fix and Resubmit	
2025/08/11 15:08	Ziyue Meng	Working On It	
2025/08/11 15:41	Ziyue Meng	Ready for Feedback	
2025/08/11 15:42	Ziyue Meng	I have upload my supplementary file, please check, thank you!	
2025/08/11 15:44	Ziyue Meng	Working On It	
2025/08/11 15:45	Ziyue Meng	Ready for Feedback	
2025/08/11 17:35	Ganesh Krishnasamy	You've NOT documented the full setup, including installing VS Code, dotnet, and SplashKit; building and running SplashKit-based programs in both C# and C++, using the terminal.	
2025/08/11 17:35	Ganesh Krishnasamy	Fix and Resubmit	
2025/08/11 17:35	Ganesh Krishnasamy	Fix and Resubmit	
2025/08/11 18:25	Ziyue Meng	Working On It	
2025/08/11 18:28	Ziyue Meng	I have already installed .NET, VSCode, and SplashKit, but I don't have screenshots. Can I just take screenshots of the statements in the terminal to check if the installation was successful? Otherwise, I'll have to uninstall everything and reinstall it and take screenshots.	
2025/08/11 19:03	Ziyue Meng	Moreover, the process of building C# and C++ programs in the terminal has been written down in the journal.	
2025/08/11 22:32	Ziyue Meng	Ready for Feedback	
2025/08/12 18:39	Ganesh Krishnasamy	Complete	

INTRODUCTION TO PROGRAMMING

ONTRACK SUBMISSION

Building and Running

Submitted By:

Ziyue MENG
zmen0034
2025/08/11 12:32

Tutor:

Ganesh KRISHNASAMY

Task Outcomes	Supports
TLO1 Document the process of setting up the programming environment and building programs, including evidence of challenges, actions, and outcomes.	ULO4 Use good programming practices
TLO2 Summarise key compilation concepts and terminal commands used when building and running programs in different languages.	ULO4 Use good programming practices
TLO3 Reflect on your learning process and describe strategies you will use to improve your confidence using the terminal.	ULO4 Use good programming practices

August 11, 2025



Build and Run

Name: Ziyue Meng
Student ID: 36035432
Date: 30/7/2025

Learning Journey and Evidence

Prior Knowledge and Experience

I once learned some python-related knowledge by myself, but I did not go very deep into it, and I have never learned the knowledge of other programming languages, nor knowledge about terminal.

Study to Acquire, Refresh, and/or Extend Capability

To study and master this topic I did the following:

28/7/2025-learned some basic command in the terminal

List files (within the current folder)

ls -l

```
ziyue@XMAS MINGW64 ~
$ ls -l
total 1
-rw-r--r-- 1 ziyue None 0 Jul 28 23:10 myeasylog.log
-rw-r--r-- 1 ziyue None 422 Jul 28 23:47 programmers-field-guide.desktop
```

↳

Move to other folders

cd /file A/file B (file B is within the file A)

```
ziyue@XMAS MINGW64 /usr/bin
$ cd /c/Users/ziyue/Documents

ziyue@XMAS MINGW64 /c/Users/ziyue/Documents
$ ls -l
total 5260
-rw-r--r-- 1 ziyue None 5382358 Mar 27 16:07 FIT1058_Asgn1_ZiyueMeng_36035432.zip..zip
lrwxrwxrwx 1 ziyue None 20 Jun 11 2024 'My Music' -> /c/Users/ziyue/Music
lrwxrwxrwx 1 ziyue None 23 Jun 11 2024 'My Pictures' -> /c/Users/ziyue/Pictures
lrwxrwxrwx 1 ziyue None 21 Jun 11 2024 'My Videos' -> /c/Users/ziyue/Videos
```

↳

Create a new folder

mkdir + folder name (the folder will be added to the folder I am currently in)

```
ziyue@XMAS MINGW64 /c/Users/ziyue/Documents
$ mkdir MyNewFolder

ziyue@XMAS MINGW64 /c/Users/ziyue/Documents
$ ls -l
total 5260
-rw-r--r-- 1 ziyue None 5382358 Mar 27 16:07 FIT1058_Asgn1_ZiyueMeng_36035432.zip..zip
lrwxrwxrwx 1 ziyue None 20 Jun 11 2024 'My Music' -> /c/Users/ziyue/Music
lrwxrwxrwx 1 ziyue None 23 Jun 11 2024 'My Pictures' -> /c/Users/ziyue/Pictures
lrwxrwxrwx 1 ziyue None 21 Jun 11 2024 'My Videos' -> /c/Users/ziyue/Videos
drwxr-xr-x 1 ziyue None 0 Jul 29 08:57 MyNewFolder

ziyue@XMAS MINGW64 /c/Users/ziyue/Documents
$ cd MyNewFolder
```

I have learned some knowledge about three main commands in the terminal.

29/7/2025-learn the basic function of terminal and shell.

The Function of Terminal and Shell

The **terminal** allows you to **input commands** and **view output**.

The **shell** has the actual logic to **interpret** and **execute** your commands.

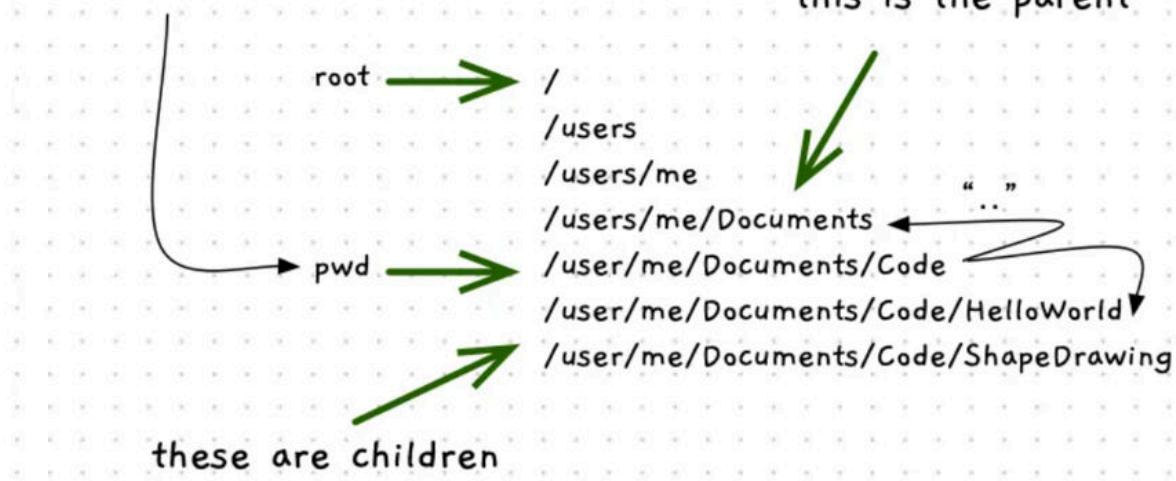
←

File Systems are Similar to A Tree

Current, Parent, Root, Child

If I am here...

this is the parent



←

Paths

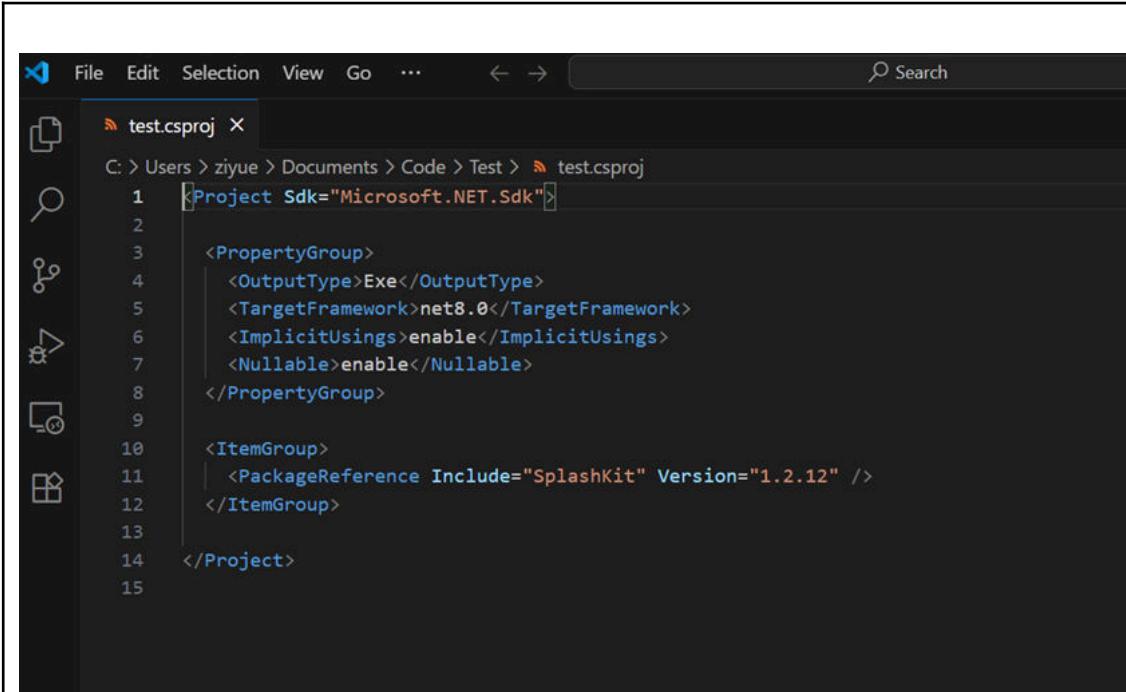
/home/multitool/Documents/code/program1.cs is a path to the file "program1.cs".

The root of the file system is the / folder, every path start with /, to indicate that

we are starting from the root.

I knew the basic structure of the folders in the computer.

28/7/2025-encounter some little problems but solve them successfully



```
<Project Sdk="Microsoft.NET.Sdk">
<PropertyGroup>
    <OutputType>Exe</OutputType>
    <TargetFramework>net8.0</TargetFramework>
    <ImplicitUsings>enable</ImplicitUsings>
    <Nullable>enable</Nullable>
</PropertyGroup>
<ItemGroup>
    <PackageReference Include="SplashKit" Version="1.2.12" />
</ItemGroup>
</Project>
```

When I first ran it, I noticed that it was different from what was shown in the Field Guide—it was missing rows 10, 11, and 12. But after I saw



```
dotnet add package SplashKit
```

Now you can use the features of [SplashKit](#) in your code.

and put them into practice, it matched what was in the Field Guide.

I knew the importance of SplashKit and the function of SplashKit.

30/7/2025-learned the dotnet programs

```
dotnet new console ---- C#
```

When running the `dotnet new console`, it will create few files:

- ◆ .vscode: is a folder to hold settings for the vscode editor we are going to use. ↵
 - ◆ program.cs: is the initial source code file for the project. ↵
 - ◆ obj: if run, using **ls obj/**, I can see that there are some files in this folder. ↵
 - ◆ Project.csproj: is a file containing settings for the project. This will be named the same as the name of the folder you run it in. ↵

Open the Folder In vscode

dotnet build ↵

compile the project in the current working directory (can be skipped). ↵

The screenshot shows a Visual Studio Code interface. The top bar displays the file path: Program.cs > Program > <top-level-statements-entry-point>. The code editor contains the following C# code:

```
1 // See https://aka.ms/new-console-template for more information
2 Console.WriteLine("Hello, World!");
3 Console.WriteLine("Enjoy the Journey");
```

The bottom navigation bar includes tabs for PROBLEMS, DEBUG CONSOLE, TERMINAL, OUTPUT, and PORTS. The TERMINAL tab is active, showing the terminal output:

```
ziyue@XMAS MINGW64 /c/users/ziyue/documents/code/test
$ dotnet run
Hello, World!
Enjoy the Journey
```

`dotnet run` ↵

build if needed, then execute your program. ↵

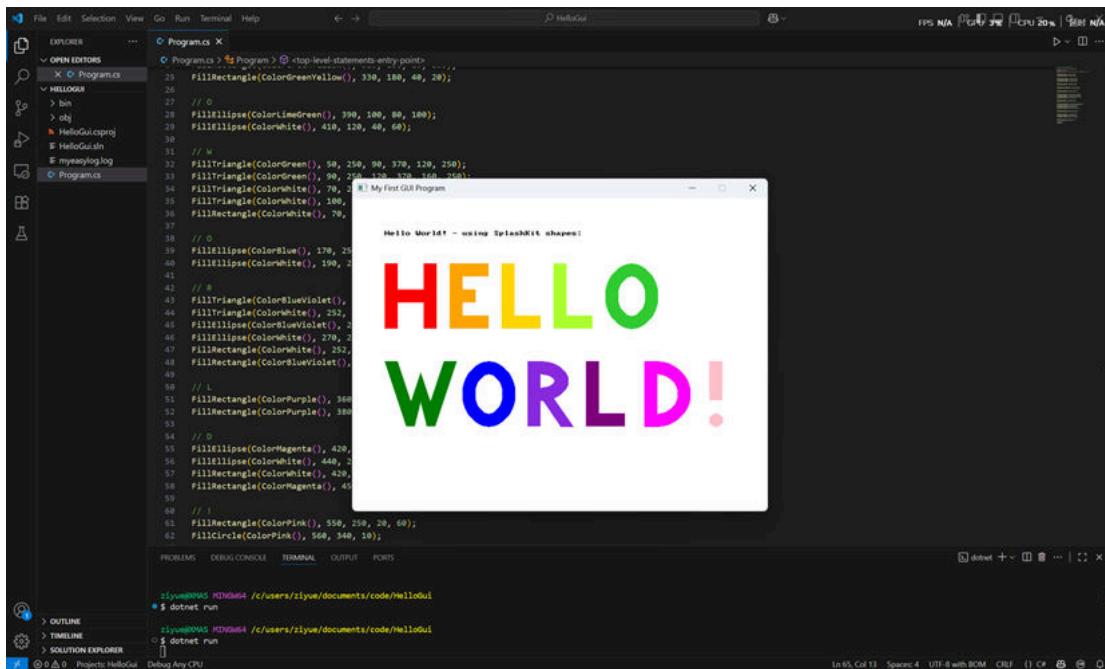
I understand the significance of the dotnet program, and I can use it to run my code, which is extremely convenient.

```
M /c/users/ziyue/documents/code/learnforp2
ziyue@XMAS MINGW64 ~
$ cd /c/users/ziyue/documents/code
ziyue@XMAS MINGW64 /c/users/ziyue/documents/code
$ mkdir learnforp2
ziyue@XMAS MINGW64 /c/users/ziyue/documents/code
$ cd learnforp2
ziyue@XMAS MINGW64 /c/users/ziyue/documents/code/learnforp2
$ dotnet new console
已成功创建模板“控制台应用”。
正在处理创建后操作...
正在还原 C:\users\ziyue\documents\code\learnforp2\learnforp2.csproj:
正在确定要还原的项目...
已还原 C:\users\ziyue\documents\code\learnforp2\learnforp2.csproj (用时 66 毫秒)。
已成功还原。

ziyue@XMAS MINGW64 /c/users/ziyue/documents/code/learnforp2
$ dotnet add package splashkit
正在确定要还原的项目...
Writing C:\msys64\tmp\pyqelre.tmp
info : X.509 证书验证选择使用 .NET 选择的默认信任存储进行代码签名。
info : X.509 证书验证选择使用 .NET 选择的默认信任存储进行时间戳。
info : 正在将包“splashkit”的 PackageReference 添加到项目“C:\users\ziyue\documents\code\learnforp2\learnforp2.csproj”。
info : GET https://api.nuget.org/v3/registration5-gz-semver2/splashkit/index.json
info : OK https://api.nuget.org/v3/registration5-gz-semver2/splashkit/index.json 250 毫秒
info : 正在还原 C:\users\ziyue\documents\code\learnforp2\learnforp2.csproj 的包...
info : GET https://api.nuget.org/v3/vulnerabilities/index.json 285 毫秒
info : GET https://api.nuget.org/v3-vulnerabilities/2025.07.31.17.40.39/vulnerability.base.json
info : GET https://api.nuget.org/v3-vulnerabilities/2025.07.31.17.40.39/2025.08.10.23.41.29/vulnerability.update.json
info : OK https://api.nuget.org/v3-vulnerabilities/2025.07.31.17.40.39/2025.08.10.23.41.29/vulnerability.update.json 276 毫秒
info : OK https://api.nuget.org/v3-vulnerabilities/2025.07.31.17.40.39/vulnerability.base.json 278 毫秒
info : 包“splashkit”与项目“C:\users\ziyue\documents\code\learnforp2\learnforp2.csproj”中指定的所有框架均兼容。
info : 包“splashkit”(版本为 1.2.12)的 PackageReference 已添加到文件“C:\users\ziyue\documents\code\learnforp2\learnforp2.csproj”。
info : 将资产文件写入磁盘。路径：C:\users\ziyue\documents\code\learnforp2\obj\project.assets.json
log  : 已还原 C:\users\ziyue\documents\code\learnforp2\learnforp2.csproj (用时 734 毫秒)。

ziyue@XMAS MINGW64 /c/users/ziyue/documents/code/learnforp2
$ code .
```

These are the methods for creating a folder and using dotnet to add the console and SplashKit libraries.



This is my HelloGui project using C# language and SplashKit library.

```

ziyue@XMAS MINGW64 /c/users/ziyue/documents/code/HelloGui
$ cd ..

ziyue@XMAS MINGW64 /c/users/ziyue/documents/code
$ mkdir CppCode

ziyue@XMAS MINGW64 /c/users/ziyue/documents/code
$ cd CppCode

ziyue@XMAS MINGW64 /c/users/ziyue/documents/code/CppCode
$ touch HelloWord.cpp

ziyue@XMAS MINGW64 /c/users/ziyue/documents/code/CppCode
$ code .

ziyue@XMAS MINGW64 /c/users/ziyue/documents/code/CppCode
$ |

```

I completed creating a folder for cpp coding.

The screenshot shows the Visual Studio Code interface. In the Explorer sidebar, there are three entries under 'OPEN EDITORS': 'HelloWorld.cpp' (selected), 'HelloWorld.cpp > main()' (highlighted in blue), and 'CPPCODE'. The main editor area displays the following C++ code:

```
#include "splashkit.h"
int main()
{
    write_line("Hello World");
    return 0;
}
```

Below the editor is the Terminal tab, which shows the command-line output of the compilation process:

```
ziyue@009AS MINGW64 /c/users/ziyue/documents/code/CppCode
$ clang++ HelloWorld.cpp -o hello -l SplashKit
./hello.exe
```

I have completed the compiling process, and I am ready to run it.

The screenshot shows the Visual Studio Code interface. In the Explorer sidebar, there are three entries under 'OPEN EDITORS': 'HelloWorld.cpp' (selected), 'HelloWorld.cpp > main()' (highlighted in blue), and 'CPPCODE'. The main editor area displays the same C++ code as before:

```
#include "splashkit.h"
int main()
{
    write_line("Hello World");
    return 0;
}
```

Below the editor is the Terminal tab, which shows the command-line output of the compilation process. However, there is an error message at the bottom:

```
clang++: error: no such file or directory: 'HelloWorld.cpp'
bash: ./hello.exe: No such file or directory
```

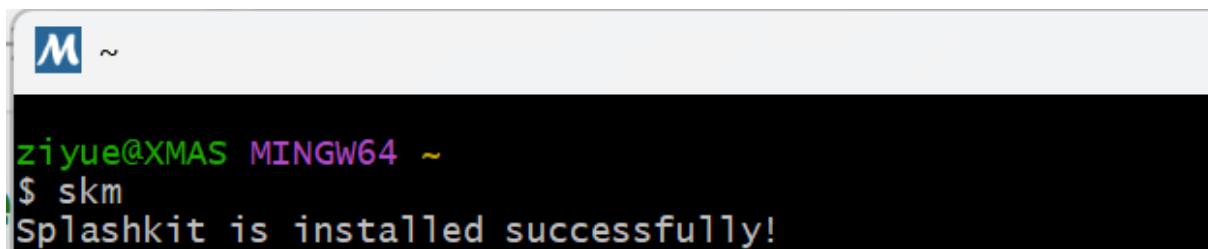
I tried to run it, but it showed that the file didn't exist. I was confused until I noticed that I had written "word" instead of "world".

```
ziyue@XMAS MINGW64 /c/users/ziyue/documents/code/CppCode
$ clang++ HelloWord.cpp -o hello -l SplashKit

ziyue@XMAS MINGW64 /c/users/ziyue/documents/code/CppCode
$ ./hello.exe
Hello World
```

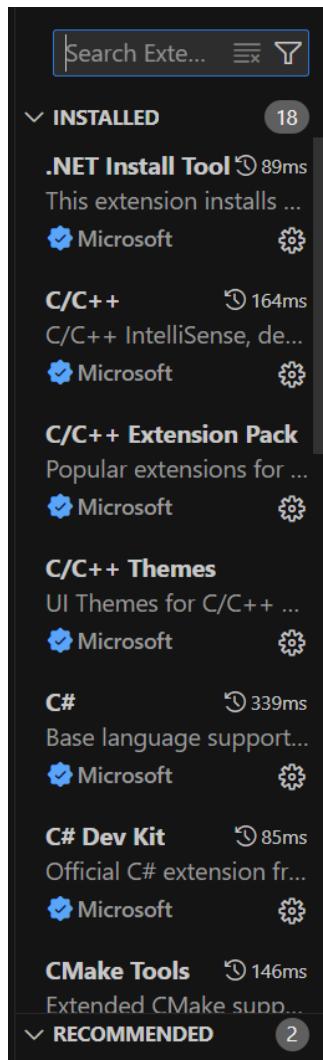
I realized the problem, corrected it, and finally got it to run successfully.

As for how to download the SplashKit and dotnet and vscode, I have done but did not make a screen shot. But when I need to check if the download is complete, I will use this code.



A screenshot of a terminal window. The title bar shows a blue 'M' icon and a tilde (~). The main area of the terminal shows the following text:

```
ziyue@XMAS MINGW64 ~
$ skm
Splashkit is installed successfully!
```



Also, I have downloaded all the tools into vscode.

Brief Summary of Concepts

Concept / Commands	Key Idea / Concept
Compiler	used to convert my code into machine code.
Shell	<ol style="list-style-type: none"> It is an interactive program to run commands. It can accept the text input and send the output to the terminal. Different shells need different commands.
Terminal	<ol style="list-style-type: none"> It is a program that I can run to interact with the computer. It can interpret my command, so that it can connect me and the shell.

Action	Example shell command
Make a folder	<code>mkdir + name</code>
Move into a folder	<code>cd /c/folder name</code>
Show/list the files in a folder	<code>ls -l</code> (I have to be in this folder)

Create a dotnet project	<code>dotnet new console</code>
Add SplashKit to a dotnet project	<code>dotnet add package SplashKit</code>
Build and run a dotnet project	<code>dotnet build</code> <code>dotnet run</code>
Compile a C/C++ program – which uses the SplashKit library	<code>clang++ HelloWorld.cpp -o hello</code> <code>-l SplashKit</code>
Run the C++ program	<code>./hello.exe</code>

Reflection

What is the most important thing you learned from this and why?

Compared with all the code I have learned, I think what's more important is that I have learned to control the computer using computer languages. This is extremely convenient and also very meaningful. Because I have developed the awareness of operating the computer with computer languages, it allows me to have a broader perspective and solve problems more calmly when I encounter some issues in the future, which will also be of great help to my study and life hereafter.

Using the terminal can be quite challenging. What strategies are you going to put in place to help you learn to use this?

Admittedly, simply learning knowledge related to code can be boring and tedious. However, when combining it with the changes happening on the computer, I can clearly see the impact my operations have on the computer, which forms a good positive feedback loop. As a result, I no longer find the code that dull—in fact, it even becomes interesting. This gives me motivation, making the learning process much easier.

6 Sequence and Data

Our first programs will involve using a sequence of instructions that manipulate data within the computer. We will focus on how the computer executes these instructions, as well as getting some more practice using the command line to build and run our programs.

Outcome		Description
TLO1		Develop simple programs using sequence, variables, and methods
TLO2		Reflect on what you've learned and evaluate personal challenges
Date	Author	Comment
2025/08/02 16:12	Ziyue Meng	Working On It
2025/08/06 13:20	Ziyue Meng	Ready for Feedback
2025/08/06 13:20	Ziyue Meng	I have done 1st and 3rd test, and I upload only 3rd test. My 1st program is shown in the Journal-reflection.
2025/08/15 15:06	Ganesh Krishnasamy	Complete

INTRODUCTION TO PROGRAMMING

ONTRACK SUBMISSION

Sequence and Data

Submitted By:

Ziyue MENG
zmen0034
2025/08/06 03:20

Tutor:

Ganesh KRISHNASAMY

Task Outcomes

- | Task Outcomes | Supports |
|--|--|
| TLO1 Develop simple programs using sequence, variables, and method calls, and explain how they work. | ULO1 Use Instructions
ULO4 Use good programming practices |
| TLO2 Reflect on what you've learned and evaluate personal challenges and strategies for programming success. | ULO4 Use good programming practices |

August 6, 2025



Sequence and Data

Name: Ziyue Meng
Student ID: 36035432
Date: 6/8/2025

Learning Journey and Evidence

Prior Knowledge and Experience

I have learned some basic commands in the terminal, such as cd, ls-l, mkdir..... And I knew how to use dotnet method to create a program, but I am not familiar with the commands and library of C Sharp at all.

Study to Acquire, Refresh, and/or Extend Capability

To study and master this topic I did the following:

<3/8/2025> - <look through the overview part>

How is the `SplashKitSDK.SplashKit` library & `System.Convert` library used? ↵

The `SplashKitSDK.SplashKit` can be used to draw graphics, play sounds, and interact with the user in the Terminal, and the `System.Convert` can be used to give me access to methods to convert data. ↵

<Through the overview section, I learned about the `SplashKitSDK.SplashKit` library and the `System.Convert` library, and now understand their purposes.>

<3/8/2025>-<Look through the overview part>

Methods and Libraries: ↵

Methods: programming languages allow you to group instructions into reusable

packages called **methods** in C#. ↵

■ Using SplashKit Library ↵

using static splashKitSDK.SplashKit; ↵

■ Using System.Convert Library ↵

using static System.Convert; ↵

↵

Method Calls and Arguments ↵

Method call: A statement that tells the computer to run a particular method. ↵

Argument: A list of values between parentheses. ↵

<I have understood what "method", "library", "method call", and "argument" mean.>

<3/8/2025>-<Look through the Overview part>

Variables, Constants

When I create a variable or constant, I must indicate the type of data(datatype).[←]

- ◆ `string _____;` ----- `“Hello”, “1045”`[←]
- ◆ `int _____;` ----- `“1”, “-20” (integer)`[←]
- ◆ `double _____;` ----- `“3.1415”, “0.333333” (real number)`[←]
- ◆ `const int _____ = _____;`[←]

This is a
constant
declaration // The circle radius is fixed


You have to give a
constant a value

Assignment statements

I can use assignment statements to store a value in the variable.[←]

- ◆ `a = b;` ----- store the value *b* in the variable *a*.[←]
- ◆ `a += b;` ----- store the value *a + b* in the variable *a*.[←]
- ◆ `a++;` ----- store the value *a + 1* in the variable *a*[←]

`a = a + 1`, `a += 1`, `a++` are the same[←]

<I have learned about the form of text and how to perform basic calculations.>

<4/8/2025>-<Learn the Simple Calculator part>

CODING(Simple Calculator)



\$\$"Hi {name}."

Can be regarded as f"Hi {name}" in python

↳

● Write(); ---- write words in the output, but do not add a enter

● WriteLine(); ---- write a line in the output, and add a enter

● a = ReadLine(); ---- store the input value in the variable a

● b = ToDouble(a); ---- convert a to double type and store it in b

**** ReadLine is regarded as string type, so we need to convert it to another**

type to operate.

↳

CODING(House Drawing)

● OpenWindow(name, width, height);

● Delay(t ms); ---- it will be around for t ms (1000ms = 1s)

<I have learned the command statements related to basic calculations in C# >

<4/8/2025>-<Coding>

The screenshot shows a Microsoft Visual Studio Code interface. The top part displays a C# file named Program.cs with line numbers. The code prompts the user for their name, distance travelled, and time taken, then calculates speed and total time for a planned distance. The bottom part shows the terminal output of the program's execution.

```
1  using static SplashKitSDK.SplashKit;
2  // using static System.Console;
3  using static System.Convert;
4  string name; //creat name variable
5  Write("What is tour name: ");
6  name = ReadLine();
7  WriteLine($"Hi, {name}.");
8
9  string userInput;
10 double distance, minutes;
11
12 Write("How far have you travelled so far(in km): ");
13 userInput = ReadLine();
14 distance = ToDouble(userInput);
15
16 Write("How long does it takes: ");
17 userInput = ReadLine();
18 minutes = ToDouble(userInput);
19
20 double answer;
21 answer = distance / (minutes/60);
22 WriteLine($"Your speed is {answer} km/h");
23
24
25 double TotalDistance, RemainingDistance;
26 double TotalTime, TimeToGo;
27
28 Write("How far do you plan to go(Enter km): ");
29 userInput = ReadLine();
30 TotalDistance = ToDouble(userInput);
31
32 RemainingDistance = TotalDistance - distance;
33 TimeToGo = RemainingDistance / answer * 60;
34 TotalTime = TimeToGo + minutes;
35
36 WriteLine($"You will take another {TimeToGo} minutes to go.");
37 WriteLine($"The total time is {TotalTime}.");
```

PROBLEMS DEBUG CONSOLE TERMINAL OUTPUT PORTS

```
How long does it takes: 50
Your speed is 6 km/h
How far do you plan to go(Enter km): 10
You will take another 50 minutes to go.
The total time is 100.

ziyue@XMAS MINGW64 /c/users/ziyue/documents/code/TravelCalculator
○ $ [ ]
```

Calculator Debug Any CPU

<This Is My First C# Work!!!!!!!!!>

<4/8/2025>-<Learn the House Drawing part>

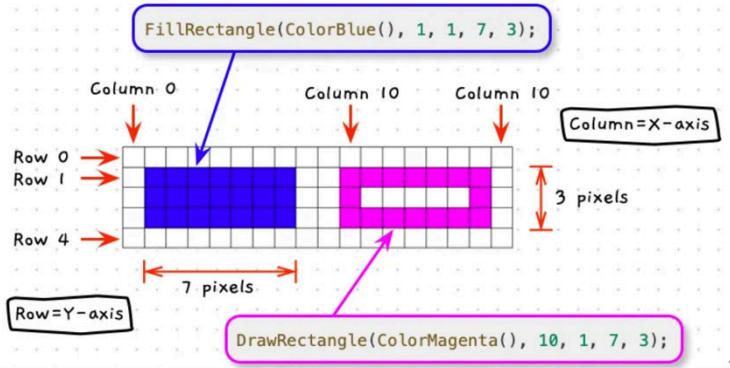
PICTURES:

- Pixels:

DrawRectangle(ColorBlue(), a, b, c, d);

FillRectangle(ColorRed(), a, b, c, d);

Window Pixels



Starting from (a, b), stretching c pixels to the right, stretching d pixels

downward.

- House Plan

ClearScreen(ColorWhite()); ----- Clear the screen

FillEllipse(ColorBrightGreen(), 0, 400, 800, 400) ----- The
coordinate of the rectangle.

RefreshScreen(); ----- Must take this step before Delay();

The order of drawing shapes affects the final image! The first down shape

is on the bottom layer, and so on.

<I have learned the command statements related to basic picture drawing in C#>

<5/8/2025>-<try to do the First task in Test Your Knowledge>

```

C# Program.cs X
C# Program.cs > Program > <top-level-statements-entry-point>
1  using static SplashKitSDK.SplashKit;
2  using static System.Convert;
3
4  string Purpose, AimDollars, PlanWeeks, HaveDollars, Capability;
5  double AimDollars_, PlanWeeks_, HaveDollars_, Capability_;
6
7  Write("What are you saving for(Enter Title): ");
8  Purpose = ReadLine();
9
10 Write("How much do you need to save(Enter Dollars): ");
11 AimDollars = ReadLine();
12 AimDollars_ = ToDouble(AimDollars);
13
14 Write("Hoe long before the purchase(Enter Weeks): ");
15 PlanWeeks = ReadLine();
16 PlanWeeks_ = ToDouble(PlanWeeks);
17
18 Write("How much do you have already(Enter Dollars): ");
19 HaveDollars = ReadLine();
20 HaveDollars_ = ToDouble(HaveDollars);
21
22 Write("How much can you save each week(Enter Dollars): ");
23 Capability = ReadLine();
24 Capability_ = ToDouble(Capability);
25
26 double NeedToSave, NeedWeeks;
27 NeedToSave = (AimDollars_ - HaveDollars_) / PlanWeeks_;
28 NeedWeeks = (AimDollars_ - HaveDollars_) / Capability_;
29
30 WriteLine($"For the {Purpose}, you need to save {NeedToSave} Dollars per week");
31 WriteLine($"Based on current savings you will need {NeedWeeks} weeks to save {AimDollars_}");

```

```

ziyue@XMAS MINGW64 /c/users/ziyue/documents/code/Calculator
$ code .

ziyue@XMAS MINGW64 /c/users/ziyue/documents/code/Calculator
$ code .

ziyue@XMAS MINGW64 /c/users/ziyue/documents/code/Calculator
$ dotnet run
What are you saving for(Enter Title): Holiday
How much do you need to save(Enter Dollars): 2572
Hoe long before the purchase(Enter Weeks): 10
How much do you have already(Enter Dollars): 2
How much can you save each week(Enter Dollars): 50
For the Holiday, you need to save 257 Dollars per week
Based on current savings you will need 51.4 weeks to save 2572

ziyue@XMAS MINGW64 /c/users/ziyue/documents/code/Calculator
$ 

```

<Without any doubt, I complete this work by myself, I think I have basically mastered the C# language related to calculator-type programs.>

<5/8/2025>-<Try on the third test in Test Your Knowledge>

```
C# Program.cs 9+ X  
C# Program.cs  
1  using SplashKitSDK;  
2  using static SplashKitSDK.SplashKit;  
3  
4  using static System.Convert;  
0 references  
5  public static SoundEffect SplashKit.LoadSoundEffect(string DoorCreak, string DoorCreaking);  
6  
7  string Name, Staff, delay;  
8  int delay_;  
9
```

<I had good progress on the picture drawing and text writing step, but when i need to add sounds, the vs code says there are errors all the time. Then I posted my question on ED Discussion, and i receive a reply by a professor, and I tried to revise my work.>

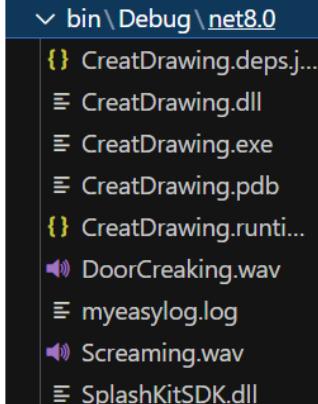
<6/8/2025>-<complete the third work in Test Your Knowledge>

```
Program.cs X
Program.cs > Program > <top-level-statements-entry-point>
1  using SplashKitSDK;
2  using static SplashKitSDK.SplashKit;
3  using static System.Convert;
4
5  SplashKit.LoadSoundEffect("DoorSound", "DoorCreaking.wav");
6  SplashKit.LoadSoundEffect("Scream", "Screaming.wav");
7
8  string Name, Staff, delay;
9  int delay_;
10
11 Write("Welcome friend. What is your name: ");
12 Name = ReadLine();
13 Write($"Hello {Name}. ");
14
15 Write("What are you most scared of: ");
16 Staff = ReadLine();
17
18 Write("Delay speed scale(1000 to 10000 ms): ");
19 delay = ReadLine();
20 delay_ =ToInt32(delay);
21
22 OpenWindow("Hunted House", 800, 600);
23 ClearScreen(ColorWhite());
24 FillEllipse(ColorBrightGreen(), 0, 400, 800, 400);
25 FillRectangle(ColorGray(), 300, 300, 200, 200);
26 FillTriangle(ColorRed(), 250, 300, 400, 150, 550, 300);
27
28 DrawText("You are approaching to a hunted house...", ColorBlack(), 150, 150);
29 RefreshScreen();
30 Delay(delay_);
31
32
33 ClearScreen(ColorBlack());
34 RefreshScreen();
35
36 SplashKit.PlaySoundEffect("DoorSound");
37
38 Delay(delay_ / 2);
39 DrawText($"{Name}, it is dark here...", ColorRed(), 500, 300);
40 RefreshScreen();
41 Delay(delay_ / 2);
42
43 ClearScreen(ColorBlack());
44
45 SplashKit.PlaySoundEffect("Scream");
46 DrawText($"{Name}!!!!", ColorRed(), "Arial", 50, 250, 300);
47 RefreshScreen();
48 Delay(delay_ / 4);
49 DrawText($"{Staff} are everywhere!!!!", ColorRed(), "Arial", 60, 240, 290);
50 RefreshScreen();
51 Delay(delay_ / 3);
```

<When I ran it, it achieved the effect I expected ! >

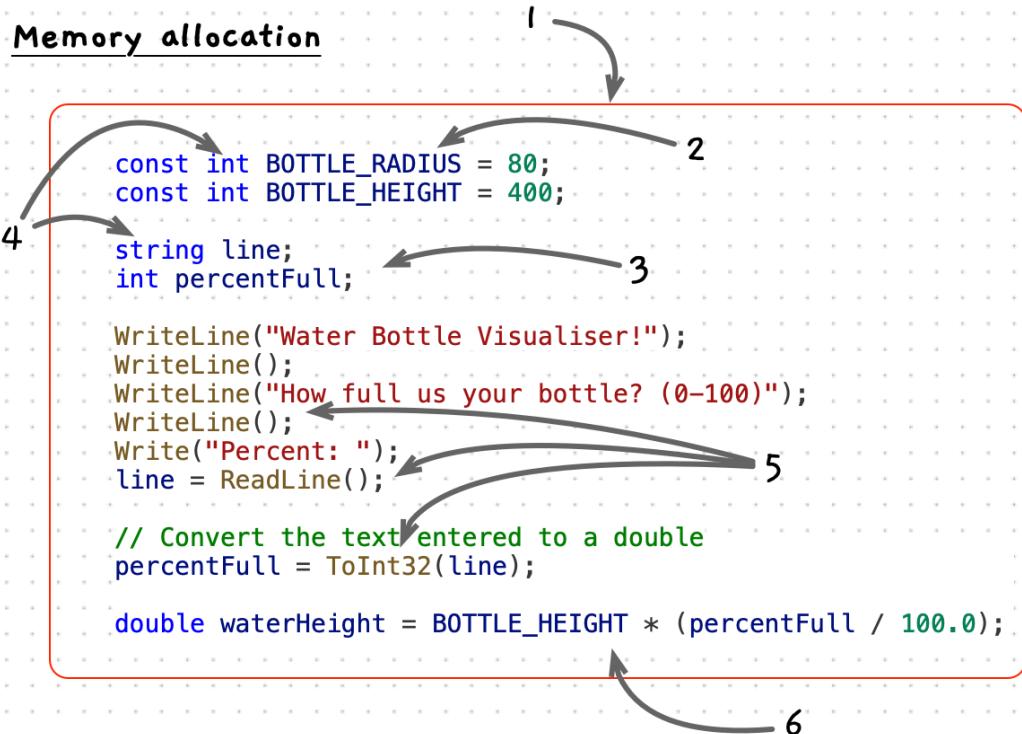
<6/8/2025>-<conclude all the main point about the test 3>

Some Important Points:

- ◆ `SplashKit.LoadSoundEffect("DoorSound", "DoorCreaking.wav");
SplashKit.LoadSoundEffect("Scream", "Screaming.wav");` is needed for loading sound file.
- ◆  Sound files are needed to save in this folder.
- ◆ `SplashKit.PlaySoundEffect("DoorSound");` This is about how to play sound.
- ◆ RefreshScreen(); should be called when we need to change the image.

Brief Summary of Concepts

Concept	Key Idea / Description
Program	A collection of code consisting of a series of statements, used to perform specific tasks.
Method	programming languages allow you to group instructions into reusable packages called methods in C#.
Variable	An identifier used to store data values, with a specific data type.
Method Call	A statement that tells the computer to run a particular method.
Assignment Statement	I can use assignment statements to store a value in the variable.
Expression	Consists of operands and operators, and generates a value after calculation.



Action	Response
1: Relate this code to the concepts of a program and sequence	This code is part of a program, and sequence refers to the code being executed in a specific order, where the statements here run sequentially in the written - order to achieve a specific function.
2: Describe BOTTLE_RADIUS and BOTTLE_HEIGHT	These are constant numbers that cannot be changed in the subsequent code.
3: Describe line and percentFull.	Line is a string variable to store the input from the console. percentFull is a variable to store converted integer value.
2+3: Explain how BOTTLE_RADIUS and percentFull use different character cases.	"BOTTLE_RADIUS" is in all uppercase, usually used to represent a constant; "percentFull" is in mixed case, used to represent a variable.
4: Describe what string and int refer to in these contexts.	String(line) is used to store the input value, because the input is string all the time. Int(BOTTLE_RADIUS, BOTTLE_HEIGHT) are constant numbers to store the radius and height of the bottle. Int(percentFull) is the integer value to store the converted integer value(percent full of the water).
5: What are WriteLine, ReadLine, andToInt32 examples of? Explain these lines of code.	WriteLine can output a line of text in the terminal, ReadLine can read the input of the terminal and store it into a certain string variable,ToInt32 is used to convert the input string to a 32-bit integer.

6: Explain the expression in this line of code and how it is evaluated.	The expression in the code "double waterHeight = BOTTLE_HEIGHT * (percentFull / 100.0); is "BOTTLE_HEIGHT * (percentFull / 100.0)". First, calculate "percentFull / 100.0" to get the fullness ratio, then multiply by "BOTTLE_HEIGHT" to get the height of the water.
---	---

Reflection

What is the most important thing you learned from this and why?

I have learned the basic programming languages in C#, including calculator-type, audio playback-type, and image generation-type programs. All of these are important, but what matters most to me is that I have understood various programming logics in C#. Moreover, by comparing it with the Python language, I have discovered many commonalities between the two and identified their respective advantages. I think this is the greatest gain because I will learn more programming languages in my future study. I need to connect all these languages and know which language to use in what situations, so that I can master computer programming.

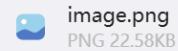
Based on what you have seen so far, what do you think your biggest challenge in this unit will be? Why? What strategies are you considering employing to overcome this?

I think the biggest challenge was definitely the code for audio loading and playback. Because when I was working on Test 3, I encountered many problems with audio, such as not knowing which folder to put the audio in. I sought help from the professor and also used AI appropriately to look up information, and finally I figured out the solution to this aspect.

References

Generative AI acknowledgement

DeepSeek—I used it to find out where audio files should be stored.



Where can i store the sound file?



To resolve the warning "**Play Sound Effect called, but no valid sound effect supplied**", you need to ensure your sound file is stored in the correct location and properly loaded. Here's how to do it:

Where to Store the Sound File

1. Default Location (Recommended)

Place the sound file (e.g., `explosion.wav`) in your project's **output directory** (where the executable runs):

- For C# (Visual Studio):

```
text  
YourProject/  
|__ bin/  
|   |__ Debug/           // Or "Release"  
|   |   |__ YourProgram.exe  
|   |   |__ explosion.wav // Store here
```

This file has additional line breaks applied by OnTrack because they contain lines longer than the configured limit. Lines over 1000 characters long have been truncated to limit PDF page count. The orginal submission can be retrieved via the "Download Uploaded Files" function.

```
1  using SplashKitSDK;
2  using static SplashKitSDK.SplashKit;
3  using static System.Convert;
4
5  SplashKit.LoadSoundEffect("DoorSound", "DoorCreaking.wav");
6  SplashKit.LoadSoundEffect("Scream", "Screaming.wav");
7
8  string Name, Staff, delay;
9  int delay_;
10
11 Write("Welcome friend. What is your name: ");
12 Name = ReadLine();
13 Write($"Hellow {Name}. ");
14
15 Write("What are you most scared of: ");
16 Staff = ReadLine();
17
18 Write("Delay speed scale(1000 to 10000 ms): ");
19 delay = ReadLine();
20 delay_ =ToInt32(delay_);
21
22 OpenWindow("Hunted House", 800, 600);
23 ClearScreen(ColorWhite());
24 FillEllipse(ColorBrightGreen(), 0, 400, 800, 400);
25 FillRectangle(ColorGray(), 300, 300, 200, 200);
26 FillTriangle(ColorRed(), 250, 300, 400, 150, 550, 300);
27
28 DrawText("You are approaching to a hunted house...", ColorBlack(), 150, 150);
29 RefreshScreen();
30 Delay(delay_);
31
32
33 ClearScreen(ColorBlack());
34 RefreshScreen();
35
36 SplashKit.PlaySoundEffect("DoorSound");
37
38 Delay(delay_ / 2);
39 DrawText($"{Name}, it is dark here...", ColorRed(), 500, 300);
40 RefreshScreen();
41 Delay(delay_ / 2);
42
43 ClearScreen(ColorBlack());
44
45 SplashKit.PlaySoundEffect("Scream");
46 DrawText($"{Name}!!!!", ColorRed(), "Arial", 50, 250, 300);
47 RefreshScreen();
48 Delay(delay_ / 4);
49 DrawText($"{Staff} are everywhere!!!!", ColorRed(), "Arial", 60, 240, 290);
50 RefreshScreen();
```

```
51 Delay(delay_ / 3);
```

7 Control Flow

Learn to use control flow statements to take your programs to the next level making them interactive and responsive. This will build upon your understanding of sequence and how the computer executes instructions, showing how you can manipulate this sequence using control flow statements.

Outcome		Description
TLO1		Apply control flow concepts (conditions and loops) to write code.
TLO2		Reflect on how sequence and control flow work together and explain how they interact.
2025/08/06 20:22	Ziyue Meng	Planned date adjusted to 15 Aug.
2025/08/06 20:22	Ziyue Meng	Working On It
2025/08/14 16:03	Ziyue Meng	Ready for Feedback
2025/08/19 14:09	Ziyue Meng	Working On It
2025/08/19 14:11	Ziyue Meng	Ready for Feedback
2025/08/20 15:52	Ganesh Krishnasamy	Your learning journey includes working programs, but there's no mention of or screenshots from the Programmers Guide. Please add this to show how you built your understanding from the course content.
2025/08/20 15:55	Ganesh Krishnasamy	Fix and Resubmit
2025/08/21 12:36	Ziyue Meng	Ready for Feedback
2025/08/21 12:38	Ziyue Meng	:grinning:I have resubmitted my work after revising, please check. Thank you for your guidance.
2025/08/21 14:09	Ganesh Krishnasamy	Complete

INTRODUCTION TO PROGRAMMING

ONTRACK SUBMISSION

Control Flow

Submitted By:

Ziyue MENG
zmen0034
2025/08/21 12:36

Tutor:

Ganesh KRISHNASAMY

Task Outcomes	Supports
TLO1 Apply control flow concepts (conditions and loops) to write correct and understandable programs.	ULO1 Use Instructions ULO4 Use good programming practices
TLO2 Reflect on how sequence and control flow work together and explain your reasoning using terminology.	ULO4 Use good programming practices



Control Flow

Name: Ziyue Meng
Student ID: 36035432
Date: 10/8/2025

Learning Journey and Evidence

Prior Knowledge and Experience

I have mastered the application of for loops, while loops, and if statements in Python, so this part of the content is not very challenging for me. I just need to learn the C# language and some other statements such as do-while.

Study to Acquire, Refresh, and/or Extend Capability

To study and master this topic I did the following:

<9/8/2025> - <learn the overview and guide tour part in filed guide>

return; means that once the return statement is executed, all the code following it will not be executed. ↵

Selection With Switch ↵

Selection with switch

The diagram shows a C# switch statement with four cases: A, a, B, b, and C, c. The default case is also shown. The first two cases (A/a) share a green box, and the last two cases (B/b and C/c) share another green box. Arrows point from the annotations to the corresponding parts of the code. The annotations are as follows:

- Switch to the case with the matching value. This will run when `line == "a" ...`
- `break` jumps to the end of the switch
- This will run when `line == "c"`
- default runs when no other branch was selected.

Break is necessary here!!!! ↵

Repeat With WhileRepetition with while

```
i = 0;
while( i < value )
{
    WriteLine("Hello World! " + i);
    i++; // i = i + 1;
}
OpenWindow("Test Window", 400, 300);
```

Loops while i is less than value

Notice we must change something from the condition (i in this case)
... otherwise when will it end?

```
while( ! QuitRequested() )
{
    FillCircle(RandomColor(), Rnd(400), Rnd(300), Rnd(50));
    RefreshScreen();
    ProcessEvents();
```

Here we loop until the user quits by closing the window...

Repeat With Do WhileRepetition with do while

```
do
{
    WriteLine("1: Say hello");
    WriteLine("2: ...");
    WriteLine("3: Quit");

    line = ReadLine();
    // ...

} while ( line != "3" );
```

Notice that in this case the instructions are before the condition, so they have to run at least once!

Loops while line does not equal "3", jumping back to the top when this needs to be repeated

< I have learned some basic statements, such as if statement, case statement, and the basic loops, I learned the basic structure of these statements. >

<10/8/2025>-<learn the in depth part in the file guide>

Difference between do-while loop and while loop:
 ↵

while Loop ↵

- Checks first, then executes:
 ↴

The condition is evaluated **before** entering the loop body.
 ↵

- If the condition is **false** initially, the loop **never runs**.
 ↵

do-while Loop ↵

- Executes first, then checks:
 ↴

The loop body runs **at least once**, then the condition is checked.
 ↵

- Even if the condition is **false** initially, the loop runs **once**.
 ↵

DIFFERENCES: ↵

Statement	Scope	Example
<code>return</code>	Exits the entire method	<code>if (x) return;</code> → Method terminates
<code>break</code>	Exits only the current loop	<code>if (x) break;</code> → Exits loop
<code>continue</code>	Skips the rest of the current iteration	<code>if (x) continue;</code> → Jumps to next iteration

Circle: ↵

`FillCircle(ColorTurquoise(), 640, 360, PLAYER_RADIUS);`
 ↵

FillCircle(Color _____(), X-coordinate, Y-coordinate, radius);
 ↵

Conditions ↵

It is vital to use brackets!!!!!!
 ↵

I understood the importance of brackets, and the differences between loops. Also, i can write some basic code about player moving in the filed guide.

<11/8/2025>-<learn the in depth part in filed guide>

```
If ()  
{  
    _____;  
    _____;  
}  
else if ()  
{  
    _____;  
    _____;  
}  
Else ()  
{  
    _____;  
}  
Switch ()  
{  
    Case "":  
        _____;  
        Break;  
    Case "":  
        _____;  
        Break;  
    Default:  
        _____;  
        Break;  
}  
}
```

```
←
```

While loop: ←

```
while (i < target)
{
    WriteLine(i);
    i++;
}
```

```
←
```

While () ←

```
{←
    _____;←
    _____;←
}←
←
←
```

Do-While loop: ←

```
do
{
    WriteLine("Hello");

    Write("Again: ");
    again = ReadLine();
} while (again == "y" || again == "Y"); ←
```

```
Do←
```

```
{←
    _____;←
    _____;←
    _____;←
} while ();←
```

```
←
```

For loop:

```
for(int i = 0; i < target; i++)  
{  
    WriteLine(i);  
}
```

For ((initialization); (condition); (iteration))

```
{  
    _____;  
}  
←
```

The differences between if and case statement:

- if is suitable for complex, non-linear conditional judgments (such as range comparisons, multi-condition combinations).
- switch-case is suitable for multi-value matching based on a single expression (direct equality judgment between values), and the code can be more concise in certain scenarios.

The differences between for, while, and do-while loop:

1. for Loop

- **Syntax:**

Combines initialization, condition, and iteration in one line.

- **Characteristics:**

1. Centralized control (initialization, condition, iteration) for clear structure.
2. Ideal when the number of iterations is known in advance.

- **Best for:**

1. Iterating over arrays, collections, or data with a defined length.
2. Fixed-repeat operations (e.g., looping from 0 to n-1).

2. while Loop

- **Syntax:**

Checks the condition first; initialization and iteration are handled externally.

- **Characteristics:**

1. Flexible conditions, not tied to a fixed iteration variable.
2. May not execute at all if the initial condition is false.

- **Best for:**

1. Scenarios where the number of iterations is unknown (e.g., waiting for user input to terminate).
2. Complex condition logic that can't be simplified to a single iteration variable.

3. do-while Loop

- **Syntax:**

Executes the loop body once first, then checks the condition.

- **Characteristics:**

1. Guarantees at least one execution of the loop body.
2. Similar to while for logic but with forced initial execution.

- **Best for:**

1. Cases requiring at least one execution (e.g., menu systems, input validation).

Differences between break, continue and goto:**break: Terminate the current loop or switch-case**

- **Scope:** Only affects loops (for/while/do-while) or switch-case statements.
- **Behavior:** Immediately terminates the current loop or switch-case, exits the code block, and proceeds to execute subsequent code.

continue: Skip current loop iteration and proceed to the next

- **Scope:** Only affects loops (for/while/do-while).
- **Behavior:** Does not terminate the entire loop; only skips code after continue in the current iteration and directly proceeds to the next iteration.

goto: Unconditionally jump to a specified label

- **Scope:** Can jump to any label within the same function (labels are defined with).
- **Behavior:** Completely ignores code structure (such as loops or condition blocks) and jumps directly to the label, which may disrupt structured program flow.
- **Note:** C# allows goto but **discourages overuse** (easily causes code confusion and poor readability), applicable only in specific scenarios (e.g., exiting nested loops).

I have consolidated the structures of all loop statements, understood the significance of indentation, and summarized the differences between different loops, as well as the differences between if and case statements, and the differences between continue, break, and goto.

I create a folder to learn the content of the programmers guide.

```
M /c/users/ziyue/documents/code/exploreforp4

ziyue@XMAS MINGW64 ~
$ cd /c/users/ziyue/documents/code/exploreforp4
ziyue@XMAS MINGW64 /c/users/ziyue/documents/code/exploreforp4
$ code .

ziyue@XMAS MINGW64 /c/users/ziyue/documents/code/exploreforp4
$
```

I have started learning about if statements.

In the Field Guide, the code he gave me starts with `using static System.Console;`, but when I wrote it out completely, there was an error. So I changed the beginning to the two most common statements I learned when studying C#, and then there were no errors.

```
1  using static System.Console;
5  string language;
6
7  Write("What language do you use? ");
8  language = ReadLine();
9  if (language == "C#")
10 {
11     WriteLine("Good choice, C# is a fine language.");
12 }
14 else if ((language == "C") || (language == "C++"))
15 {
16     WriteLine("These are great languages.");
17 }
18 else
19 {
20     WriteLine("Well... good luck with that!");
21 }
22
23 WriteLine("Great chat!");
```

The screenshot shows the Visual Studio Code interface. The code editor displays the C# program above. The file explorer on the left shows a project named 'EXPLOREFORP4' containing files like 'bin', 'obj', 'exploreforp4.csproj', 'exploreforp4.sln', and 'myeasylog.log'. The terminal at the bottom shows the output of running the program:

```
ziyue@XMAS MINGW64 /c/users/ziyue/documents/code/exploreforp4
$ dotnet run
What language do you use? Python
Well... good luck with that!
Great chat!
```

The terminal also shows the command being run: '\$ dotnet run'.

I have started learning about case statements.

The screenshot shows the Visual Studio Code interface. On the left is the Explorer sidebar with a project named 'EXPLOREFORP4' containing files like 'bin', 'obj', 'exploreforp4.csproj', 'exploreforp4.sln', and 'myeasylog.log'. The 'Program.cs' file is selected and shown in the main editor area. The code contains a switch statement:

```
using static System.Convert;
string language;
Write("What language do you use? ");
language = ReadLine();
switch (language)
{
    case "C#":
        WriteLine("Good choice, C# is a fine language.");
        break;
    case "C":
    case "C++":
        WriteLine("These are great languages.");
        break;
    default:
        WriteLine("Well... good luck with that!");
        break;
}
WriteLine("Great chat!");
```

Below the editor are tabs for PROBLEMS, DEBUG CONSOLE, TERMINAL, OUTPUT, and PORTS. The TERMINAL tab is active, showing command-line history:

```
What language do you use? Python
Well... good luck with that!
Great chat!

ziyue@XMAS MINGW64 /c/users/ziyue/documents/code/exploreforp4
$ dotnet run \
> ^C

ziyue@XMAS MINGW64 /c/users/ziyue/documents/code/exploreforp4
$ dotnet run
What language do you use? C#
Good choice, C# is a fine language.
Great chat!

ziyue@XMAS MINGW64 /c/users/ziyue/documents/code/exploreforp4
$
```

No comprehension issues arose, and the code was successfully run.

I have started learning about `while loop`.

```
C# Program.cs > Program > <top-level-statements>
  2  using static System.Convert;
  54 int target;
  55
  56 Write("Count to: ");
  57 target =ToInt32(ReadLine());
  58
  59 int i = 0;
  60
  61 while (i < target)
  62 {
  63     WriteLine(i);
  64     i++;
  65 }
```

```
ziyue@XMAS MINGW64 /c/users/ziyue/documents/code/exploreforp4
● $ dotnet run
Count to: 6
0
1
2
3
4
5
```

Since I have learned these statements in the Python language, learning loops and conditional statements in C# hasn't posed much difficulty for me.

DO-WHILE:

```
2     using static System.Convert;
3
72     string again;
73
74     do
75     {
76         WriteLine("Hello");
77
78         Write("Again: ");
79         again = ReadLine();
80     } while (again == "y" || again == "Y");
```

```
ziyue@XMAS MINGW64 /c/users/ziyue/documents/code/exploreforp4
● $ dotnet run
Hello
Again: y
Hello
Again: y
Hello
Again: n
```

FOR LOOP:

```
2  using static System.Convert;
88
89  int target;
90
91  Write("Count to: ");
92  target =ToInt32(ReadLine());
93
94  for(int i =  0; i < target; i++)
95  {
96      WriteLine(i);
97  }
```

```
ziyue@XMAS MINGW64 /c/users/ziyue/documents/code/explorefor4
● $ dotnet run
Count to: 8
0
1
2
3
4
5
6
7
```

Whole process of the test 1:

```

ziyue@XMAS MINGW64 ~
$ cd /c/users/ziyue/documents/code
ziyue@XMAS MINGW64 /c/users/ziyue/documents/code
$ mkdir SimpleStats
ziyue@XMAS MINGW64 /c/users/ziyue/documents/code
$ cd SimpleStats
ziyue@XMAS MINGW64 /c/users/ziyue/documents/code/SimpleStats
$ dotnet new console
已成功创建模板“控制台应用”。

正在处理创建后操作...
正在还原 C:\users\ziyue\documents\code\SimpleStats\SimpleStats.csproj:
正在确定要还原的项目...
已还原 C:\users\ziyue\documents\code\SimpleStats\SimpleStats.csproj (用时 46 毫秒)。
已成功还原。

ziyue@XMAS MINGW64 /c/users/ziyue/documents/code/SimpleStats
$ dotnet add package splashkit
正在确定要还原的项目...
Writing C:\msys64\temp\tmphmztjq.tmp
info : X.509 证书链验证将使用 .NET 选择的默认信任存储进行代码签名。

```

I complete the process of creating the folder and add splashkit in the terminal.

```

Program.cs X
Program.cs > Program > <top-level-statements-entry-point>
1  using static SplashKitSDK.SplashKit;
2  using static System.Convert;
3
4  WriteLine("Welcome to the simple stats calculate! ");
5  WriteLine("");
6
7  string userInput;
8  int value, count, total, min, max, average;
9
10 do
11 {
12     WriteLine("Enter a value: (input \"exit\" to stop.) ");
13     userInput = ReadLine();
14     count = 1;
15     value =ToInt32(userInput);
16     total =ToInt32(userInput);
17     min =ToInt32(userInput);
18     max =ToInt32(userInput);
19     average =ToInt32(userInput);
20     WriteLine($"count: {count}, value: {value}, total: {total}, min: {min}, max: {max}, average: {average}");
21 } while (userInput != "exit" || userInput != "Exit");

```

The main framework has been completed.

Only some issues in the code need to be fixed now.

I realized that I only need to store the value of userInput once.

```
20      min = number;
21      max = number;
22      if (count == 1)
23      {
24          min = number;
25          max = number;
26      }
27      else
28      {
29          if (number >= max)
30          {
31              Max = number;
32          }
33          else
34          {
35              Max = max;
36          }
37          if (number <= min)
38          {
39              Min = min;
40          }
41      }
```

I used nested if loops to implement the method for comparing sizes.

```
20     min = number;
21     max = number;
22     if (count == 1)
23     {
24         Min = number;
25         Max = number;
26     }
27     else
28     {
29         if (number >= max)
30         {
31             Max = number;
32         }
33         else
34         {
35             Max = max;
36         }
37         if (number <= min)
38         {
39             Min = number;
40         }
41         else
42         {
43             Min = min;
44         }
45     }
46 }
```

```
Enter a value: (input "exit" to stop.)
10
count: 2, value: 10, total: 10, min: 10, max: 10, average: 5
Enter a value: (input "exit" to stop.)
5
count: 2, value: 5, total: 5, min: 5, max: 5, average: 2
Enter a value: (input "exit" to stop.)
3
count: 2, value: 3, total: 3, min: 3, max: 3, average: 1
```

It didn't meet my expectations, so I checked my code and found that there were problems with the code related to the count counter. Then I changed to using if statements to judge the quantity.

Also, I found that compared to while loops, which can directly control whether to enter the loop through initial conditions, do-while cannot skip the loop body through

conditions before the first execution. Therefore, in scenarios where "it may not execute even once" is required (such as deciding whether to start a process based on parameters), do-while is not as intuitive as while.

So I changed the code to the while loop.

```
int count = 0;
count++;
Count = count;
```

I initialized count and then used a while loop.

```
18    while (userInput != "exit")
19    {
20        count++;
21        Count = count;
22    }
```

```
5    if (count == 1)
5    {
6        min = number;
7        max = number;
8    }
9    else
10    {
11        if (number > max)
12        {
13            max = number;
14        }
15        if (number < min)
16        {
17            min = number;
18        }
19    }
```

But I found that there was a problem in the update process of min and max. Because I added `min = number` and `max = number` before the if statement, the values of min and max were always numbers.

I fixed this by removing Max and Min, and also removing the else part in the inner if loop.

```
14  while (userInput != "exit")
15  {
```

Final optimization part: When the condition of my while loop was like this, I had to type something myself to start the loop before writeline appeared. But I wanted the loop to run automatically and exit automatically when I typed "exit".

```
while (true)
{
    WriteLine("Enter a value: (input \"exit\" to stop.) ");
    userInput = ReadLine();
    if (userInput == "exit")
    {
        break;
}
```

So I used while (true) to make the loop run automatically, and added an if statement later to check if my input was "exit". If it was, the loop would exit.

FINAL OUTPUT:

```

ziyue@XMAS MINGW64 /c/users/ziyue/documents/code/simplestats
● $ dotnet run
Welcome to the simple stats calculate!

Enter a value: (input "exit" to stop.)
10
count: 1, value: 10, total: 10, min: 10, max: 10, average: 10
Enter a value: (input "exit" to stop.)
20
count: 2, value: 20, total: 30, min: 10, max: 20, average: 15
Enter a value: (input "exit" to stop.)
15
count: 3, value: 15, total: 45, min: 10, max: 20, average: 15
Enter a value: (input "exit" to stop.)
5
count: 4, value: 5, total: 50, min: 5, max: 20, average: 12.5
Enter a value: (input "exit" to stop.)
exit

ziyue@XMAS MINGW64 /c/users/ziyue/documents/code/simplestats
○ $ █

```

COMPLETE!!!!!!

Brief Summary of Concepts

Concept / Commands	Key Idea / Concept
Boolean Data	It is used when we need to judge true or false in a certain sentence, and the output will be true or false.
Comparison	Comparisons are a common way of generating a boolean value in your code. These expressions allow you to compare two values and determine the truth of a given condition.
Logical Operators	A and B(A && B) : if and only if both A and B are true, the output would be true. A or B(A B) : if and only if both A and B are false, the output would be false, if not, the output will be true. A xor B(A ^ B) : if and only if A and B are not the same, the output will be true, if not, the output will be false. not A(! A) : if A is true, not A is false.
And	A and B(A && B) : if and only if both A and B are true, the output would be true.
Or	A or B(A B) : if and only if both A and B are false, the output would be false, if not, the output will be true.
Not	not A(! A) : if A is true, not A is false.
If statement	A conditional statement that executes a block of code if a specified condition is true.
Switch case statement	A multi - way branching statement. It tests the value of a variable and jumps to the corresponding case block.

While loop	A loop that repeatedly executes a block of code as long as a specified condition is true. <pre> 10 ~ while (!QuitRequested()) 11 { 12 ProcessEvents(); 13 ~ if (KeyDown(KeyCode.RightKey)) 14 { 15 x += MOVE_SPEED; 16 } 17 ~ if (KeyDown(KeyCode.LeftKey)) 18 { 19 x -= MOVE_SPEED; 20 } 21 ~ if (KeyTyped(KeyCode.SpaceKey)) 22 { 23 clr = RandomColor(); 24 } 25 ClearScreen(ColorWhite()); 26 FillCircle(clr, x, 300, 50); 27 RefreshScreen(60); 28 29 </pre>
Do while loop	Similar to a while loop, but it first executes the code block once and then checks the condition.
For loop	A loop that is often used when you know the number of iterations in advance. It has an initialization, a condition, and an increment/decrement part.
Break	A statement used to terminate the execution of a loop (such as a for, while, or do - while loop) or a switch statement prematurely and jump out of the nearest enclosing loop or switch.
Continue	A statement that skips the remaining statements in the current iteration of a loop (for, while, do - while) and proceeds to the next iteration.
Indentation	In C#, curly braces {} are used to represent code - blocks. Indentation is used to improve code readability and help visually group related statements together.
Go to	Unconditionally transfers the control flow to a labeled statement in the program. However, its use is often discouraged because it can make the code difficult to understand and maintain.

Reflection

What is the most important thing you learned from this and why?

Learning loop statements (for, while, do-while) and conditional statements (if, switch-case) has not only enabled me to master the ability to handle batch problems with concise logic—where a single loop can replace hundreds or thousands of repetitive operations—but also endowed programs with "decision-making capabilities" through conditional statements, allowing flexible responses to varied inputs and multiple scenarios. The combined use of these basic statements has let me experience the essence of programming as "creating rules," advancing from "understanding code" to "expressing ideas with code." I deeply feel the power and convenience of programming in simplifying complex tasks through logic and transforming abstract requirements into concrete implementations, while also developing structured thinking skills.

How do you feel sequence and control flow are connected? Does understanding sequence help you understand control flow?

Sequence and control flow are inherently intertwined in programming. The sequence represents the default, linear order in which statements are executed, from top - to - bottom in a program. Control flow, on the other hand, is what allows us to deviate from this strict sequence. For example, conditional statements like if - else and loops such as for and while break the monotony of the sequential execution.

Understanding sequence is fundamental to grasping control flow. When we know the normal order of execution, we can better appreciate how control flow statements change that order. For instance, in a for loop, the sequence of operations within the loop body is executed repeatedly, but this repetition is a controlled deviation from the normal one - time - through sequence. Without a solid understanding of sequence, it would be difficult to make sense of how control flow constructs modify the overall execution path of a program. In short, sequence serves as the foundation upon which the more complex concept of control flow is built.

Why is indentation so important?

Indentation is crucial in programming. It clearly presents the subordination relationships of code blocks through a visual hierarchical structure. The logical levels of structures such as conditional statements and loops are immediately apparent. Whether in single - person development or team collaboration, it can significantly enhance code readability and maintainability, helping developers quickly understand the execution order and dependency relationships and avoid logical errors caused by structural chaos.

References

Generative AI acknowledgement

I use Grammarly to check my grammar and correct my mistakes.

This file has additional line breaks applied by OnTrack because they contain lines longer than the configured limit. Lines over 1000 characters long have been truncated to limit PDF page count. The orginal submission can be retrieved via the "Download Uploaded Files" function.

```
1  using static SplashKitSDK.SplashKit;
2  using static System.Convert;
3
4  WriteLine("Welcome to the simple stats calculate! ");
5  WriteLine(" ");
6
7  string userInput;
8  int min = 0, max = 0, number;
9  double average;
10 int Total;
11 int count = 0;
12 int total = 0;
13 while (true)
14 {
15     WriteLine("Enter a value: (input \"exit\" to stop.) ");
16     userInput = ReadLine();
17     if (userInput == "exit")
18     {
19         break;
20     }
21     number =ToInt32(userInput);
22
23     count++;
24     Total = total += number;
25     average = (double)Total / count;
26
27     if (count == 1)
28     {
29         min = number;
30         max = number;
31     }
32     else
33     {
34         if (number > max)
35         {
36             max = number;
37         }
38         if (number < min)
39         {
40             min = number;
41         }
42     }
43
44     WriteLine($"count: {count}, value: {number}, total: {Total}, min: {min}, max:
45     ↵ {max}, average: {average}");
46 }
```

8 C/C++

The next part of our journey will explore how to organise our code and data. To do this, we need to first switch programming languages. We will be using C/C++ for this next part and this task focuses on the similarities and differences you will find with this new language.

Outcome	Description		
Date	Author	Comment	
TLO1		Understand how existing programming concepts such as control structures work in C/C++.	
TLO2		Recognise how compiled languages build and link with external libraries.	
TLO3		Reflect on key C++ concepts and compare with C#, documentation and examples.	
2025/08/06 20:23	Ziyue Meng	Planned date adjusted to 22 Aug.	
2025/08/14 16:11	Ziyue Meng	Working On It	
2025/08/17 18:14	Ziyue Meng	Ready for Feedback	
2025/08/19 14:15	Ziyue Meng	Ready for Feedback	
2025/09/01 11:25	Ganesh Krishnasamy	Hi Ziyue.	
2025/09/01 11:25	Ganesh Krishnasamy	There are no comments in your code. Please revise and add documentation to show your intent and help others read it.	
2025/09/01 11:26	Ganesh Krishnasamy	Also, please include white spaces between the main blocks of code as a good programming practice.	
2025/09/01 11:27	Ganesh Krishnasamy	Your Learning Journey and Evidence section lists some actions but doesn't always explain what you learned from them. Add more detail about the outcomes of your study.	
2025/09/01 11:27	Ganesh Krishnasamy	Fix and Resubmit	
2025/09/01 13:52	Ziyue Meng	Ready for Feedback	
2025/09/01 13:53	Ziyue Meng	I have modified my work and reupload it, thank you for your guidance!	
2025/09/05 12:42	Ganesh Krishnasamy	Complete	

INTRODUCTION TO PROGRAMMING

ONTRACK SUBMISSION

C/C++

Submitted By:

Ziyue MENG
zmen0034
2025/09/01 13:52

Tutor:

Ganesh KRISHNASAMY

Task Outcomes

- TLO1 Understand how existing programming concepts such as control flow structures work in C++ and transfer from C#.
- TLO2 Recognise how compiled languages build and link with external libraries.
- TLO3 Reflect on key C++ concepts and compare with C#, documenting learning clearly.

Supports

- ULO1 Use Instructions
- ULO2 Programs, Functions, and Types
- ULO3 Use libraries
- ULO4 Use good programming practices

September 1, 2025



C/C++

Name: Ziyue Meng
Student ID: 36035432
Date: 17/8/2025

Learning Journey and Evidence

Prior Knowledge and Experience

I have basically mastered some statements in the C# language and can skillfully use loops, but I know nothing about C++. I don't know the differences between it and C#, nor do I know how to write a C++ code.

Study to Acquire, Refresh, and/or Extend Capability

To study and master this topic I did the following:

<15/8/2025> - <learn the method to open and compile the program from Field Guide>

The screenshot shows the Visual Studio Code interface. In the Explorer sidebar, there is a folder named 'MYCPPCODE' containing files 'hello.cpp', 'hello.exe', and 'myeasylog.log'. The 'hello.cpp' file is open in the main editor area, displaying the following code:

```
#include "splashkit.h"
int main()
{
    write_line("Hello World! ");
    return 0;
}
```

The terminal tab is active, showing the following command-line session:

```
ziyue@XMAS MINGW64 /c/users/ziyue/documents/code1/MyCppCode
$ clang++ hello.cpp -l SplashKit -o hello
ziyue@XMAS MINGW64 /c/users/ziyue/documents/code1/MyCppCode
$ ./hello
Hello World!

ziyue@XMAS MINGW64 /c/users/ziyue/documents/code1/MyCppCode
$
```

The terminal window has a title bar 'M /c/users/ziyue/documents/code1/MyCppCode'.

I knew how to open a Cpp file and compile the program, which is quite different from C#.

The screenshot shows a code editor window with the file `hello.cpp` open. The code is as follows:

```
hello.cpp 2 X
hello.cpp > main()
1 #include "splashkit.h"
2
3 int main()
4 {
5     write_line("testing method calls");
6     int delay_seconds;
7     string user_input;
8
9     write("How many seconds do you want it to wait for: Enter seconds: ");
10    user_input = read_line();
11    delay_seconds = stoi(user_input);
12    open_window("Test window", 1280, 720);
13    clear_screen(random_color());
14    refresh_screen();
15    delay(delay_seconds * 1000);
16    return 0;
17 }
```

Below the code editor, the VS Code interface shows the following tabs: PROBLEMS (2), DEBUG CONSOLE, TERMINAL, OUTPUT, and PORTS. The PROBLEMS tab is active, displaying two errors:

- identifier "string" is undefined C/C++(20) [Ln 7, Col 5]
- identifier "stoi" is undefined C/C++(20) [Ln 11, Col 21]

I encountered a problem and posted it on ed. The professor answered my question, but there are still several tasks left to be done.

```
↳ hello.cpp > abc stoi
1 #include "splashkit.h"
2
3 using std::stoi;
4 using std::to_string;
5
6 int main()
7 {
8     string name, line;
9     int age;
10
11    write("Enter your name: ");
12    name = read_line();
13
14    write_line("Hello " + name + "!");
15
16    write("How old are you? Enter your age: ");
17
18    line = read_line();
19    while( ! is_integer(line) )
20    {
21        write_line("Sorry, I cannot convert that to a number");
22        write("How old are you? Enter your age: ");
23        line = read_line();
24    }
25
26    age = stoi(line);
27
28    write_line("You are " + to_string(age) + "!");
29
30    return 0;
31 }
```

I tried the code in the field guide again and found there were still errors. I asked everyone on ed, and they said my code was fine. So I continued to compile and run it, and finally successfully got the result.

```

ziyue@XMAS MINGW64 /c/users/ziyue/documents/code1/mycppcode
$ clang++ hello.cpp -l splashkit -o hello

ziyue@XMAS MINGW64 /c/users/ziyue/documents/code1/mycppcode
$ ./hello
Enter your name: ziyue
Hello ziyue!
How old are you? Enter your age: 19
You are 19!

ziyue@XMAS MINGW64 /c/users/ziyue/documents/code1/mycppcode
$ 

```

I started on the converting work

1. Comment out all C# code

```

EXPLORER      ...
OPEN EDITORS  x hello.cpp
CONVERTINGCODE  x hello.cpp

C: hello.cpp  x
1 // using static SplashKitSDK.SplashKit;
2 // using static System.Convert;
3
4 // WriteLine("Welcome to the simple stats calculate! ");
5 // WriteLine(" ");
6
7 // string userInput;
8 // int min = 0, max = 0, number;
9 // double average;
10 // int Total;
11 // int count = 0;
12 // int total = 0;
13 // while (true)
14 // {
15 //     WriteLine("Enter a value: (input \"exit\" to stop.) ");
16 //     userInput = ReadLine();
17 //     if (userInput == "exit")
18 //     {
19 //         break;
20 //     }
21 //     number =ToInt32(userInput);
22
23 //     count++;
24 //     Total = total += number;
25 //     average = (double)Total / count;
26
27 //     if (count == 1)
28 //     {
29 //         min = number;
30 //         max = number;
31 //     }
32 //     else
33 //     {
34 //         if (number > max)
35 //         {
36 //             max = number;
37 //         }
38 //         if (number < min)
39 //         {
40 //             min = number;
41 //         }
42 //     }
43
44 //     WriteLine($"count: {count}, value: {number}, total: {Total}, min: {min}, max: {max}, average: {average}");
45 // }

```

2. Add the required libraries and add the framework

The screenshot shows a code editor interface with a dark theme. On the left, there's an 'EXPLORER' sidebar with sections for 'OPEN EDITORS' (containing 'hello.cpp') and 'CONVERTINGCODE' (also containing 'hello.cpp'). Below these are 'OUTLINE' and 'TIMELINE' sections. The main area is titled 'hello.cpp' and contains the following C++ code:

```
4 int main()
5 {
6     // WriteLine("Welcome to the simple stats calculate! ");
7     // WriteLine(" ");
8
9
10    // string userInput;
11    // int min = 0, max = 0, number;
12    // double average;
13    // int Total;
14    // int count = 0;
15    // int total = 0;
16    // while (true)
17    //{
18        // WriteLine("Enter a value: (input \"exit\" to stop.) ");
19        // userInput = ReadLine();
20        if (userInput == "exit")
21        {
22            break;
23        }
24        number =ToInt32(userInput);
25
26        count++;
27        Total = total += number;
28        average = (double)Total / count;
29
30        if (count == 1)
31        {
32            min = number;
33            max = number;
34        }
35        else
36        {
37            if (number > max)
38            {
39                max = number;
40            }
41            if (number < min)
42            {
43                min = number;
44            }
45        }
46
47        WriteLine($"count: {count}, value: {number}, total: {Total}, min: {min}, max: {max}, average: {average}");
48    }
49    return 0;
50 }
```

3. The processing of parameters and data has been completed.

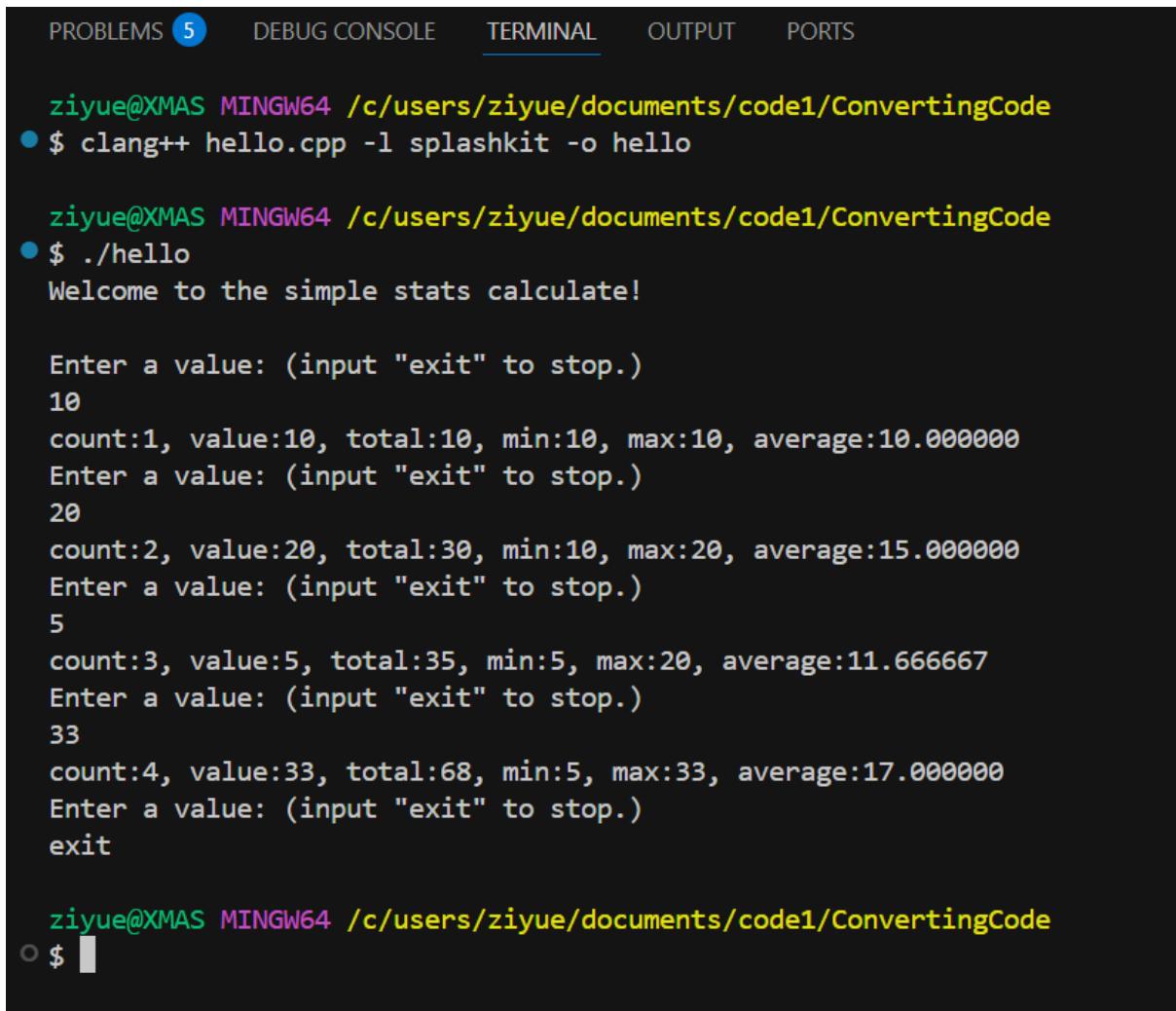
```
1 // using static SplashKitSDK.SplashKit;
2 // using static System.Convert;
3 #include "splashkit.h"
4 using std::stoi;
5 using std::to_string;
6
7 int main()
8 {
9 // WriteLine("Welcome to the simple stats calculate! ");
10 // WriteLine(" ");
11
12 string userInput;
13 int min = 0, max = 0, number;
14 double average;
15 int Total;
16 int count = 0;
17 int total = 0;
18 // while (true)
19 // {
20 //     WriteLine("Enter a value: (input \"exit\" to stop.)");
21 //     userInput = ReadLine();
22 //     if (userInput == "exit")
23 //     {
24 //         break;
25 //     }
26     number = stoi(userInput);
27 }
```

4. The conversion of command formats from C# to C++ has been completed.

```
↳ hello.cpp 6 X
↳ hello.cpp > ⚙ main()
1 // using static SplashKitSDK.SplashKit;
2 // using static System.Convert;
3 #include "splashkit.h"
4 using std::stoi;
5 using std::to_string;
6
7 int main()
8 {
9     write_line("Welcome to the simple stats calculate! ");
10    write_line(" ");
11
12    string userInput;
13    int min = 0, max = 0, number;
14    double average;
15    int Total;
16    int count = 0;
17    int total = 0;
18    while (true)
19    {
20        write_line("Enter a value: (input \"exit\" to stop.) ");
21        userInput = read_line();
22        if (userInput == "exit")
23        {
24            break;
25        }
26        number = stoi(userInput);
27
28        count++;
29        Total = total += number;
30        average = (double)Total / count;
31
32        if (count == 1)
33        {
34            min = number;
35            max = number;
36        }
37        else
38        {
39            if (number > max)
40            {
41                max = number;
42            }
43            if (number < min)
44            {
45                min = number;
46            }
47        }
    }
```

```
        }
    write_line("count:" + to_string(count) + ", value:" + to_string(number) + ", total:" + to_string(Total) + ", min:" + to_string(min) + ", max:" + to_string(max) + ",
}
return 0;
```

5. compile and run!



```
PROBLEMS 5 DEBUG CONSOLE TERMINAL OUTPUT PORTS

ziyue@XMAS MINGW64 /c/users/ziyue/documents/code1/ConvertingCode
● $ clang++ hello.cpp -l splashkit -o hello

ziyue@XMAS MINGW64 /c/users/ziyue/documents/code1/ConvertingCode
● $ ./hello
Welcome to the simple stats calculate!

Enter a value: (input "exit" to stop.)
10
count:1, value:10, total:10, min:10, max:10, average:10.000000
Enter a value: (input "exit" to stop.)
20
count:2, value:20, total:30, min:10, max:20, average:15.000000
Enter a value: (input "exit" to stop.)
5
count:3, value:5, total:35, min:5, max:20, average:11.666667
Enter a value: (input "exit" to stop.)
33
count:4, value:33, total:68, min:5, max:33, average:17.000000
Enter a value: (input "exit" to stop.)
exit

ziyue@XMAS MINGW64 /c/users/ziyue/documents/code1/ConvertingCode
○ $
```

SUCCESSFUL!

<19/8/2025>

I received guidance from the professor.



Charlotte Pierce STAFF

1小时前



Try this!



1. Run `gcc -v -E -x c++ -` in a terminal
2. In the output, find the line `#include <...> search starts here:`
3. Copy every path under that, until you get to `End of search list.`
4. Paste those paths into the VS code settings file, in the `"C_Cpp.default.systemIncludePath":` section. Add quotes around each path, and a comma at the end of each line to format the JSON correctly

This should sort out VS Code struggling to find the libraries. Let a TA know in class if you need help with this!

评论 ...

He asked me to follow a series of steps.

M ~

```

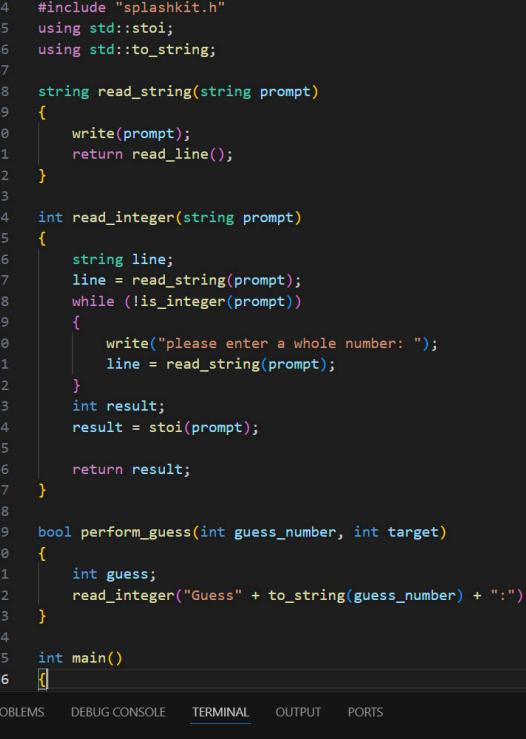
ziyue@XMAS MINGW64 ~
$ gcc -v -E -x c++ -
Using built-in specs.
COLLECT_GCC=C:\msys64\mingw64\bin\gcc.exe
Target: x86_64-w64-mingw32
Configured with: ../gcc-15.1.0/configure --prefix=/mingw64 --with-local-prefix=/mingw64/local --with-native-system-header-dir=/mingw64/include --libexecdir=/mingw64/lib --enable-bootstrap --enable-checking=release --with-arch=nocona --with-tune=generic --enable-mingw-wildcard --enable-languages=c,c++,fortran,ada,objc,obj-c++,objc++ --enable-shared --enable-static --enable-libatomic --enable-threadsafe-posix --enable-graphite --enable-fuzzy-string --enable-libstdcxx-backtrace=yes --enable-libstdcxx-filesystem-ts --enable-libstdcxx-time --disable-libstdcxx-pch --enable-lto --enable-libgomp --disable-libssp --disable-multilib --disable-rpath --disable-win32-registry --disable-nls --disable-error-warning --disable-symvers --with-libiconv --with-system-zlib --with-gmp=/mingw64 --with-mpfr=/mingw64 --with-mpc=/mingw64 --with-isl=/mingw64 --with-pkgversion='Rev8, Built by MSYS2 project' --with-build-url=https://github.com/msys2/MINGW-packages/issues --with-gnu-as --with-gnu-ld --with-libstdcxx-zeninfo=yes --disable-libstdcxx-debug --enable-plugin --with-boot-ldflags=-static-libstdc++ --with-stage1-ldflags=-static-libstdc+
Thread model: posix
Supported LTO compression algorithms: zlib zstd
gcc version 15.1.0 (Rev8, Built by MSYS2 project)
COLLECT_GCC_OPTIONS='-v' '-E' '-mtune=generic' '-march=nocona'
  C:/msys64/mingw64/bin/../lib/gcc/x86_64-w64-mingw32/15.1.0/cc1plus.exe -E -quiet -v -iprefix C:/msys64/mingw64/bin/../lib/gcc/x86_64-w64-mingw32/15.1.0/ -D_REENTRANT - -mtune=generic -march=nocona -dUMPbase -
ignoring nonexistent directory "C:/msys64/mingw64/bin/../lib/gcc/x86_64-w64-mingw32/15.1.0/../../../../x86_64-w64-mingw32/include"
ignoring duplicate directory "C:/msys64/mingw64/lib/gcc/../../../../lib/gcc/x86_64-w64-mingw32/15.1.0/../../../../include/c++/15.1.0"
ignoring duplicate directory "C:/msys64/mingw64/lib/gcc/../../../../lib/gcc/x86_64-w64-mingw32/15.1.0/../../../../include/c++/15.1.0/x86_64-w64-mingw32"
ignoring duplicate directory "C:/msys64/mingw64/lib/gcc/../../../../lib/gcc/x86_64-w64-mingw32/15.1.0/../../../../include/c++/15.1.0/backward"
ignoring duplicate directory "C:/msys64/mingw64/lib/gcc/../../../../lib/gcc/x86_64-w64-mingw32/15.1.0/../../../../include/c++/15.1.0/include"
ignoring nonexistent directory "D:/M/msys64/mingw64/include"
ignoring nonexistent directory "/mingw64/include"
ignoring duplicate directory "C:/msys64/mingw64/lib/gcc/../../../../lib/gcc/x86_64-w64-mingw32/15.1.0/include-fixed"
ignoring nonexistent directory "C:/msys64/mingw64/lib/gcc/../../../../lib/gcc/x86_64-w64-mingw32/15.1.0/../../../../x86_64-w64-mingw32/include"
ignoring nonexistent directory "D:/M/msys64/mingw64/include"
#include "..." search starts here:
#include <...> search starts here:
  C:/msys64/mingw64/bin/../lib/gcc/x86_64-w64-mingw32/15.1.0/../../../../../../../../include/c++/15.1.0
  C:/msys64/mingw64/bin/../lib/gcc/x86_64-w64-mingw32/15.1.0/../../../../../../../../include/c++/15.1.0/x86_64-w64-mingw32
  C:/msys64/mingw64/bin/../lib/gcc/x86_64-w64-mingw32/15.1.0/../../../../../../../../include/c++/15.1.0/backward
  C:/msys64/mingw64/bin/../lib/gcc/x86_64-w64-mingw32/15.1.0/include
  C:/msys64/mingw64/bin/../lib/gcc/x86_64-w64-mingw32/15.1.0/../../../../../../../../include
  C:/msys64/mingw64/bin/../lib/gcc/x86_64-w64-mingw32/15.1.0/include-fixed
End of search list.

```

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows a folder named "RUN AND DEBUG..." containing files like ".vscode", "CMakeLists.txt", "hello.cpp", and "main.cpp".
- Search Bar:** Displays "learnforp6".
- Left Sidebar:** Includes icons for File, Edit, Selection, View, Go, Run, Terminal, Help, and a Run and Debug button.
- Central Area:** The "Settings" tab is selected, showing the contents of the "settings.json" file. The file path is "C:\Users\ziyue\AppData\Roaming\Code\User\workspaceStorage\1\settings.json". The code defines profiles for terminals and environments, including paths for Mingw64 and MinGW-w64 compilers.
- Bottom Navigation:** Includes tabs for PROBLEMS, DEBUG CONSOLE, TERMINAL, OUTPUT, and PORTS.
- Terminal:** Shows the command line at the bottom left: "ziyue@XMAS MINGW64 /c/users/ziyue/documents/code1/learnforp6".
- Status Bar:** Shows line 46, column 2; spaces: 4; CRLF; and JSON with Comments.

I did as instructed, and the code returned to normal. All error messages are gone!!!



```
zhiyue@XMAS MINGW64 /c/users/zhiyue/documents/code1/learnforp6
$
```

Brief Summary of Concepts

Concept	Key Idea / Concept
Program entry point	The starting position where the program begins to execute
Main function	The entry - function of program execution. The program starts to execute code in sequence from here. In C++ programs, it is usually the "main" function.
Linker	A program that, in the final stage of the compilation process, combines object files and library files into an executable file and resolves external symbol references.
Naming conventions	Rules and habits for naming identifiers (such as variables, functions, classes, etc.) in programming, which help improve code readability and maintainability.
Different capitalisation formats (eg. Pascal, Camel, Snake, Upper)	snake_case for functions and procedures (the equivalent of C# methods). snake_case for variables. UPPER_CASE for constants.

Action	Example shell command to do action
Compile a C/C++ program with the SplashKit library	<code>clang++ hello.cpp -l SplashKit -o hello</code>
Run the C++ program	<code>./hello</code>

Reflection

What is the most important thing you learned from this and why?

The most important thing I've learned is the role of different programming elements. Take naming conventions, for example. Giving appropriate names makes the code clear at a glance, and greatly simplifies subsequent maintenance and modifications. For instance, naming variables with meaningful names allows people to immediately understand what they represent, just like putting clear labels on rooms makes it easy to find things. Understanding these helps me write more standardized and readable code, which is very beneficial for improving my programming skills.

What were the main similarities you noticed between C# and C/C++?

They all have similar syntax structures. For example, they all use curly braces {} to define code blocks, such as function bodies and loop bodies. Moreover, there are overlaps in variable declarations and data types. They all have common data types like integers and floating - point numbers. In terms of control - flow statements, for instance, if - else conditional statements, for and while loops, the usage is also quite similar. They are all used to control the execution flow of the program.

References

Generative AI acknowledgement

I used GenAI to check my language expression and grammar errors.

This file has additional line breaks applied by OnTrack because they contain lines longer than the configured limit. Lines over 1000 characters long have been truncated to limit PDF page count. The orginal submission can be retrieved via the "Download Uploaded Files" function.

```
1 #include "splashkit.h" //This is the method to use labrary Splashkit in c++.
2 using std::stoi;
3 using std::to_string;
4 // i must write these two lines before using `stoi` and `to_string`.
5
6 int main()
7 {
8     write_line("Welcome to the simple stats calculate! "); // The writing method of
9     // → `write_line` is fidderent from that in C#.
10    write_line(" ");
11
12    string userInput;
13    int min = 0, max = 0, number;
14    double average;
15    int Total;
16    int count = 0;
17    int total = 0;
18    while (true)
19    {
20        write_line("Enter a value: (input \"exit\" to stop.) ");
21        userInput = read_line();
22        if (userInput == "exit")
23        {
24            break;
25        }
26        number = stoi(userInput);
27
28        count++;
29        Total = total += number;
30        average = (double)Total / count; // using (double) before the variable can
31        // → get the value more precise.
32
33        if (count == 1)
34        {
35            min = number;
36            max = number;
37        }
38        else
39        {
40            if (number > max)
41            {
42                max = number;
43            }
44            if (number < min)
45            {
46                min = number;
47            }
48        }
49    }
50 }
```

```
48     write_line("count:" + to_string(count) + ", value:" + to_string(number) + ",  
49     ↵ total:" + to_string(Total) + ", min:" + to_string(min) + ", max:" +  
50     ↵ to_stri  
51     ng(max) + ", average:" + to_string(average));  
52     // if i need to write a sentence in C++ using write_line, i need to use "+"  
53     ↵ symbol to connect different elements.  
54 }  
55 return 0;  
56 }
```

9 Structuring Data

Now that you have functions and procedures to help you organise your code, you can start to build larger programs. One problem at the moment is how to organise the data in your program. Simple variables are getting clunky, so we need to look at ways we can start to organise our data more meaningfully in our code.

Outcome		Description	
Date	Author	Comment	
TLO1		Demonstrate engagement with the learning journey by documenting your work.	
TLO2		Demonstrate understanding of structs, enums, and fields, and how they are used.	
TLO3		Apply structs and enums to implement a program that organises data.	
2025/08/06 20:23	Ziyue Meng	Planned date adjusted to 29 Aug.	
2025/08/24 17:19	Ziyue Meng	Working On It	
2025/08/28 17:28	Ziyue Meng	Ready for Feedback	
2025/09/05 20:17	Ganesh Krishnasamy	Krish-	Hi Ziyue.
2025/09/05 20:17	Ganesh Krishnasamy	Krish-	There are no comments in your code. Please revise and add documentation to show your intent and help others read it.
2025/09/05 20:22	Ganesh Krishnasamy	Krish-	I can't see any engagement with the guided tour and concepts for Structring Data. At least, engagement with one of the guided tour and capture the learning journey.
2025/09/05 20:22	Ganesh Krishnasamy	Krish-	Fix and Resubmit
2025/09/06 12:49	Ziyue Meng	Ready for Feedback	
2025/09/06 16:03	Ziyue Meng	I have modified my work and reupload it, thank you for your guidance!	
2025/09/09 09:10	Ganesh Krishnasamy	Krish-	You have included many screenshots of your learning, but it would be better if you can describe what you have learned from doing all these that shown in the screenshots.
2025/09/09 09:10	Ganesh Krishnasamy	Krish-	Fix and Resubmit
2025/09/09 14:03	Ziyue Meng	Ready for Feedback	
2025/09/09 14:03	Ziyue Meng	I have modified my work and reupload it, thank you for your guidance!	
2025/09/17 16:27	Ganesh Krishnasamy	Krish-	Complete

INTRODUCTION TO PROGRAMMING

ONTRACK SUBMISSION

Structuring Data

Submitted By:

Ziyue MENG
zmen0034
2025/09/09 14:03

Tutor:

Ganesh KRISHNASAMY

Task Outcomes	Supports
TLO1 Demonstrate engagement with the learning journey by documenting actions, evidence, and reflection on progression.	ULO4 Use good programming practices
TLO2 Demonstrate understanding of structs, enums, and fields, and reflect on their role in larger programs.	ULO2 Programs, Functions, and Types ULO4 Use good programming practices
TLO3 Apply structs and enums to implement a program that organises data appropriately.	ULO2 Programs, Functions, and Types ULO3 Use libraries

September 9, 2025



Organising Data

Name: Ziyue Meng
Student ID: 36035432
Date: 28/8/2025

Learning Journey and Evidence

To study and master this topic I did the following:

```
M /c/users/ziyue/documents/code1/learnforp7

ziyue@XMAS MINGW64 ~
$ cd /c/users/ziyue/documents/code1

ziyue@XMAS MINGW64 /c/users/ziyue/documents/code1
$ mkdir learnforp7

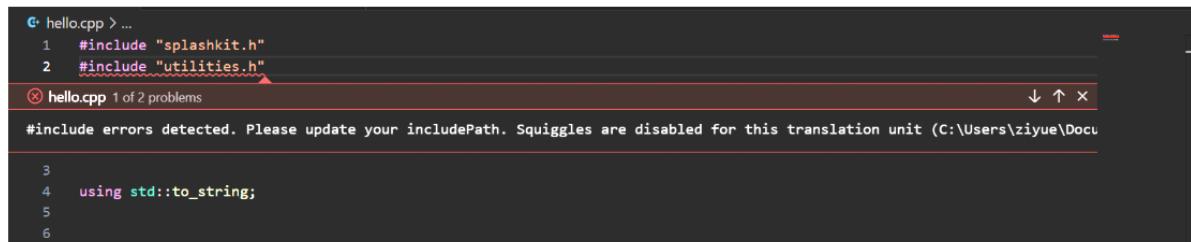
ziyue@XMAS MINGW64 /c/users/ziyue/documents/code1
$ cd learnforp7

ziyue@XMAS MINGW64 /c/users/ziyue/documents/code1/learnforp7
$ touch hello.cpp

ziyue@XMAS MINGW64 /c/users/ziyue/documents/code1/learnforp7
$ code .

ziyue@XMAS MINGW64 /c/users/ziyue/documents/code1/learnforp7
$
```

I have created a file for learning P7.



```
hello.cpp > ...
1 #include "splashkit.h"
2 #include "utilities.h"
@ hello.cpp 1 of 2 problems
#include errors detected. Please update your includePath. Squiggles are disabled for this translation unit (C:\Users\ziyue\Docu
3
4     using std::to_string;
5
6
```

I encountered this problem where utilities.h would throw an error, so I sought help on ed. Someone told me that I need to create a utilities.cpp file and a utilities.h file.

```

ziyue@XMAS MINGW64 ~
$ cd /c/users/ziyue/documents/code1/learnforp7

ziyue@XMAS MINGW64 /c/users/ziyue/documents/code1/learnforp7
$ touch utilities.cpp

ziyue@XMAS MINGW64 /c/users/ziyue/documents/code1/learnforp7
$ touch utilities.h

ziyue@XMAS MINGW64 /c/users/ziyue/documents/code1/learnforp7
$ code .

ziyue@XMAS MINGW64 /c/users/ziyue/documents/code1/learnforp7
$
```

The problem has been solved!!!!

The writing of the struct form has been completed.

Struct

This declares a struct called book_data

Each book data variable will have two field: title and pages

int main()

book_data book_0 = {"Happy Days", 20};
 book_data book_1;
 book_data book_2;

```

struct book_data
{
    string title;
    int pages;
};
```

This declares three book_data variables - each with a title and pages.

//... This code can be used to initialise the fields of a struct. This sets the title to "Happy Days" and pages to 20,

Annotated book_data struct and example variables

From this image, I've learned about structs in programming. A struct named "book_data" is declared, which contains two fields: a string - type "title" and an int -

type "pages". Each variable of the "book_data" struct will have these two fields. In the "main" function, three variables of type "book_data" are declared, with "book_0" being initialized with the title "Happy Days" and 20 pages. This shows how to define a struct and how to initialize its variables.

```
6  ✓ struct book_data
7    {
8      string title;
9      int pages;
10     };
11
12 ✓ void print_book(book_data book)
13   {
14     write_line(book.title + "(" + to_string(book.pages) + "pages)");
15   }
16
17 |
```

The basic functions have been completed.

```
✓ string read_string(string prompt)
{
  write(prompt);
  return read_line();
}

✓ int read_integer(string prompt)
{
  write(prompt);
  string line = read_line();
  return stoi(line);
}
```

```
book_data read_book(string prompt)
{
    book_data result = {"", 0};
    write_line(prompt);
    result.title = read_string("Enter book title: ");
    result.pages = read_integer("Enter number of pages: ");
    return result;
}
```

```
40  int main()
41  {
42      book_data book_0 = {"Happy days", 20};
43      book_data book_1;
44      book_data book_2;
45
46      book_1 = read_book("Enter book 1 details.");
47      book_2 = read_book("Enter book 2 details.");
48
49      print_book(book_0);
50      print_book(book_1);
51      print_book(book_2);
52
53      return 0;
54 }
55
```

Learning these snippets has been rewarding. The "read_book" function shows structs in practice: it takes input and returns a "book_data" struct with title and pages, demonstrating data encapsulation. The "main" function illustrates variable declaration—some direct, others via "read_book"—clarifying usage and user-input handling. These examples teach struct syntax and designing functions for custom types, key for organized software.

```
ziyue@XMAS MINGW64 /c/users/ziyue/documents/code1/learnforp7
● $ clang++ hello.cpp -l splashkit -o hello

ziyue@XMAS MINGW64 /c/users/ziyue/documents/code1/learnforp7
● $ ./hello
Enter book 1 details.
Enter book title: FIT1045
Enter number of pages: 100
Enter book 2 details.
Enter book title: ADS1002
Enter number of pages: 200
Happy days(20pages)
FIT1045(100pages)
ADS1002(200pages)

ziyue@XMAS MINGW64 /c/users/ziyue/documents/code1/learnforp7
○ $ █
```

SUCCESSFUL ! ! !

Enum

This declares an enum called genre_data

The list of values for each genre variable

Each value maps to an integer, from 0 to 5 in this case

```
enum genre_data
```

```
{
```

```
FANTASY = 0,  
SCIENCE_FICTION,  
MYSTERY,  
ROMANCE,  
COOKING,  
NONFICTION
```

```
};
```

Create a test_genre variable to store a genre_data value

```
int main()
```

```
{
```

```
// Initialise book_0  
genre_data test_genre = MYSTERY;
```

MYSTERY is a genre_data value

From this image, I've further expanded my programming knowledge to enums. An enum named "genre_data" is declared, which includes a set of named constants representing different book genres: FANTASY (assigned the value 0), SCIENCE_FICTION, MYSTERY, ROMANCE, COOKING, and NONFICTION. Each value in this enum implicitly maps to an integer, starting from 0 and incrementing for each subsequent constant. In the "main" function, a variable "test_genre" of type "genre_data" is created and initialized with the value "MYSTERY", demonstrating how to define and use an enum variable to store one of the predefined genre-related values.

```
64 enum genre_data
65 {
66     FANTASY = 0,
67     SCIENCE_FICTION,
68     MYSTERY,
69     ROMANCE,
70     COOKING,
71     NONFICTION
72 };
73
74 const int GENRE_COUNT = (int)NONFICTION + 1;
75
76 string to_string(genre_data genre)
77 {
78     switch(genre)
79     {
80         case FANTASY:
81             return "fantasy";
82         case SCIENCE_FICTION:
83             return "science fiction";
84         case MYSTERY:
85             return "mystery";
86         case ROMANCE:
87             return "romance";
88         case COOKING:
89             return "cooking";
90         case NONFICTION:
91             return "nonfiction";
92         default:
93             return "unknow";
94     }
95 }
96
```

Complete Learning ENUM ! ! ! !

Then I started to code for 'TEST YOUR KNOWLEDGE' part.

```
1 #include "splashkit.h"
2 #include "utilities.h"
3
4 using std::to_string;
5 using std::stoi;
```

The basic steps have been completed.

```
int read_integer(string prompt)
{
    write(prompt);
    string line = read_line();
    return stoi(line);
}

string read_string(string prompt)
{
    write(prompt);
    return read_line();
}
```

It seems that this step is necessary in most C++ code.

```
20 enum menu_option
21 {
22     READ_BOOK = 1,
23     PRINT_BOOK,
24     QUIT
25 };
```

I have created this enumeration structure.

```
26  
27     struct book_data  
28     {  
29         string title;  
30         string location;  
31         int pages;  
32     };
```

The book_data has been created according to the requirements of the problem.

```
34  ↘ menu_option read_menu_option()  
35  {  
36      write_line("\nMenu:");  
37      write_line("1 - Read book");  
38      write_line("2 - Print book");  
39      write_line("3 - Quit");  
40  
41      int choice = read_integer("Option: ");  
42      return (menu_option)choice;  
43  }
```

Define a new function using menu_option, and use (menu_option)choice to forcefully convert the int-type choice to the menu_option form.

```
45  ↘ book_data read_book_data()  
46  {  
47      book_data book;  
48  
49      write_line("Book entry system:\n");  
50  
51      book.title = read_string("Enter the name of the book: ");  
52      book.location = read_string("Enter book location: ");  
53      book.pages = read_integer("Enter number of pages: ");  
54  
55      return book;  
56  }  
57
```

```
58 ˜ void print_book_details(const book_data book)
59  {
60      write_line("\nBook details:");
61      write_line("Title: " + book.title);
62      write_line("Location: " + book.location);
63      write_line("Pages: " + to_string(book.pages));
64      write_line();
65  }
66
67 ˜ void read_book(const book_data book)
68  {
69      write_line("Opening book: " + book.title);
70      write_line("Accessing location: " + book.location);
71      write_line("This book has " + to_string(book.pages) + " pages.");
72      write_line("Reading simulation complete.\n");
73 }
```

The construction of the function model for the main part has been completed.

```
58 ˜ void print_book_details(const book_data book)
59  {
60      write_line("\nBook details:");
61      write_line("Title: " + book.title);
62      write_line("Location: " + book.location);
63      write_line("Pages: " + to_string(book.pages));
64      write_line();
65  }
66
67 ˜ void read_book(const book_data book)
68  {
69      write_line("Opening book: " + book.title);
70      write_line("Accessing location: " + book.location);
71      write_line("This book has " + to_string(book.pages) + " pages.");
72      write_line("Reading simulation complete.\n");
73 }
```

```
● 76 ✘ int main()
77     {
78         book_data my_book;
79         menu_option option;
80
81         my_book = read_book_data();
82
83     do
84     {
85         option = read_menu_option();
86
87     switch(option)
88     {
89         case READ_BOOK:
90             read_book(my_book);
91             break;
92
93         case PRINT_BOOK:
94             print_book_details(my_book);
95             break;
96
97         case QUIT:
98             write_line("Goodbye!");
99             break;
100
101     default:
102         write_line("Invalid option!");
103         break;
104     }
105
106     } while (option != QUIT);
107
108     return 0;
109 }
```

The writing of the main part has been completed.

```
ziyue@XMAS MINGW64 /c/users/ziyue/documents/code1/p7work
● $ clang++ hello.cpp utilities.cpp -l splashkit -o hello

ziyue@XMAS MINGW64 /c/users/ziyue/documents/code1/p7work
● $ ./hello
Book entry system:

Enter the name of the book: Programmers Guide
Enter book location: \\programmers guide
Enter number of pages: 500

Menu:
1 - Read book
2 - Print book
3 - Quit
Option: 2

Book details:
Title: Programmers Guide
Location: \\programmers guide
Pages: 500

Menu:
1 - Read book
2 - Print book
3 - Quit
Option: 1
Opening book: Programmers Guide
Accessing location: \\programmers guide
This book has 500 pages.
Reading simulation complete.

Menu:
1 - Read book
2 - Print book
3 - Quit
Option: 3
Goodbye!

ziyue@XMAS MINGW64 /c/users/ziyue/documents/code1/p7work
○ $ █
```

SUCCESSFUL!!!!!!

Brief Summary of Concepts

Concept	Key Idea / Concept
Enum	A collection of named constants that improves code readability, e.g., representing menu options.
Struct	A composite type that combines variables of different types to organize related data, e.g., book information (title, author, etc.).
Field	An individual variable within a struct, e.g., the "title" field in a book struct.

Reflection

What is the most important thing you learned from this and why?

Mastering structs is crucial because they enable efficient organization of related data. In large-scale projects, multi-type data (such as user information like name and age) is much easier to manage with structs.

How do you see structs helping you create larger programs? Relate this to things you have created during this task.

Structs can model real-world entities. For example, in a library program, a book struct can store details like title, making it convenient to handle batches of books and enhancing code modularity and reusability.

Looking back at what you have learnt so far, how are you going sticking to your original learning plan?

I'm mostly on track, though I spent more time than planned on complex uses of structs. This adjusted my schedule a bit but was well worth it.

How does this unit compare with other units you're currently taking?

It focuses more on basic data structures, laying the foundation for data organization needed in other courses (e.g., algorithms or design patterns).

How are you tracking toward achieving your target grade?

I'm on schedule. I've completed all tasks, can apply what I've learned in practice, and am on track to reach my target grade.

References

Generative AI acknowledgement

No Gen AI be used in this work.

This file has additional line breaks applied by OnTrack because they contain lines longer than the configured limit. Lines over 1000 characters long have been truncated to limit PDF page count. The orginal submission can be retrieved via the "Download Uploaded Files" function.

```
1 #include "splashkit.h"
2 #include "utilities.h"
3
4 using std::to_string;
5 using std::stoi;
6
7
8 // Function read_integer: Reads an integer based on a prompt
9 // Parameter prompt: Message displayed to the user as a prompt
10 // Return value: Integer input by the user
11 int read_integer(string prompt)
12 {
13     write(prompt);
14     string line = read_line();
15     return stoi(line);
16 }
17
18 // Function read_string: Reads a string based on a prompt
19 // Parameter prompt: Message displayed to the user as a prompt
20 // Return value: String input by the user
21 string read_string(string prompt)
22 {
23     write(prompt);
24     return read_line();
25 }
26
27 // Define an enumeration type menu_option to represent menu choices
28 enum menu_option
29 {
30     READ_BOOK = 1,
31     PRINT_BOOK,
32     QUIT
33 };
34
35 // Define a structure book_data to store book-related information
36 struct book_data
37 {
38     string title;
39     string location;
40     int pages;
41 };
42
43 // Function read_menu_option: Reads the user's menu selection
44 menu_option read_menu_option()
45 {
46     write_line("\nMenu:");
47     write_line("1 - Read book");
48     write_line("2 - Print book");
49     write_line("3 - Quit");
50 }
```

```
51     int choice = read_integer("Option: ");
52     return (menu_option)choice;
53 }
54
55 // Function read_book_data: Reads detailed information about a book
56 book_data read_book_data()
57 {
58     book_data book;
59
60     write_line("Book entry system:\n");
61
62     book.title = read_string("Enter the name of the book: ");
63     book.location = read_string("Enter book location: ");
64     book.pages = read_integer("Enter number of pages: ");
65
66     return book;
67 }
68
69 // Function print_book_details: Displays detailed information about a book
70 void print_book_details(const book_data book)
71 {
72     write_line("\nBook details:");
73     write_line("Title: " + book.title);
74     write_line("Location: " + book.location);
75     write_line("Pages: " + to_string(book.pages));
76     write_line();
77 }
78
79 // Function read_book: Simulates opening and reading a book
80 void read_book(const book_data book)
81 {
82     write_line("Opening book: " + book.title);
83     write_line("Accessing location: " + book.location);
84     write_line("This book has " + to_string(book.pages) + " pages.");
85     write_line("Reading simulation complete.\n");
86 }
87
88
89 int main()
90 {
91     book_data my_book;
92     menu_option option;
93
94     my_book = read_book_data();
95
96     do
97     {
98         option = read_menu_option();
99
100        switch(option)
101        {
102            case READ_BOOK:
103                read_book(my_book);
104                break;
```

```
105
106     case PRINT_BOOK:
107         print_book_details(my_book);
108         break;
109
110     case QUIT:
111         write_line("Goodbye!");
112         break;
113
114     default:
115         write_line("Invalid option!");
116         break;
117     }
118 } while (option != QUIT);
119
120 return 0;
121 }
122 }
```

10 Test 1

Programming is a foundational skill in computing, and it is important to understand this well before progressing to other units. The tests provide a means of you demonstrating you have achieved the required knowledge and skills.

Date	Author	Comment
2025/08/29 17:53	Ziyue Meng	Ready for Feedback
2025/09/02 18:11	Ganesh Krishnasamy	Fix and Resubmit
2025/09/04 17:43	Juan Du	Complete

11 Indirect Access

References and pointers allow you to access other locations in memory - giving you indirect access to data within memory. In this task you will learn to use references and pointers to allow you to create procedures that update passed in values, and use pointers to set up relationships between entities.

Outcome		Description	
Date	Author	Comment	
TLO1	Ziyue Meng	Demonstrate engagement with the learning journey by documenting your progress.	
TLO2	Ziyue Meng	Explain and apply the concepts of references and pointers, illustrating how they work.	
TLO3	Ziyue Meng	Apply indirect access by constructing a working program that swaps two variables.	
2025/08/06 20:23	Ziyue Meng	Planned date adjusted to 29 Aug.	
2025/08/19 23:39	Ziyue Meng	Working On It	
2025/08/24 17:19	Ziyue Meng	Not Started	
2025/08/28 14:01	Ziyue Meng	Working On It	
2025/09/03 14:13	Ziyue Meng	Ready for Feedback	
2025/09/17 16:09	Ganesh Krishnasamy	Hi Ziyue.	
2025/09/17 16:09	Ganesh Krishnasamy	You have included many screenshots of your learning, but it would be better if you can describe what you have learned from doing all these that shown in the screenshots.	
2025/09/17 16:14	Ganesh Krishnasamy	Fix and Resubmit	
2025/09/17 16:29	Ziyue Meng	Okay, I will fix that. Thank you.	
2025/09/17 16:41	Ziyue Meng	Ready for Feedback	
2025/09/17 16:41	Ziyue Meng	I have modified my work and reupload it, thank you for your guidance!	
2025/09/24 16:22	Ganesh Krishnasamy	Pls use enum for mapping the switch case.	
2025/09/24 16:22	Ganesh Krishnasamy	Fix and Resubmit	
2025/09/29 13:35	Ziyue Meng	Ready for Feedback	
2025/10/01 13:18	Ganesh Krishnasamy	You should build a small test program to illustrate swapping of the two variables.	
2025/10/01 13:18	Ganesh Krishnasamy	Fix and Resubmit	
2025/10/01 15:31	Ziyue Meng	Ready for Feedback	
2025/10/01 15:31	Ziyue Meng	I have modified my work and reupload it, thank you for your guidance!	
2025/10/04 20:46	Ganesh Krishnasamy	Complete	

INTRODUCTION TO PROGRAMMING

ONTRACK SUBMISSION

Indirect Access

Submitted By:

Ziyue MENG
zmen0034
2025/10/01 15:31

Tutor:

Ganesh KRISHNASAMY

Task Outcomes	Supports
TLO1 Demonstrate engagement with the learning journey by documenting study actions, evidence, and reflection on progression and coding practice.	ULO2 Programs, Functions, and Types ULO4 Use good programming practices
TLO2 Explain and apply the concepts of references and pointers, illustrate their behaviour through accurate hand execution, and analyse procedure calls and variable interactions in written responses.	ULO2 Programs, Functions, and Types ULO4 Use good programming practices
TLO3 Apply indirect access by constructing a working program that updates and interacts with values using references and pointers, and evaluate the implications for modular program design.	ULO1 Use Instructions ULO2 Programs, Functions, and Types ULO3 Use libraries ULO4 Use good programming practices

October 1, 2025



Indirect Access

Name: Ziyue Meng
Student ID: 36035432
Date: 3/9/2025

Learning Journey and Evidence

Prior Knowledge and Experience

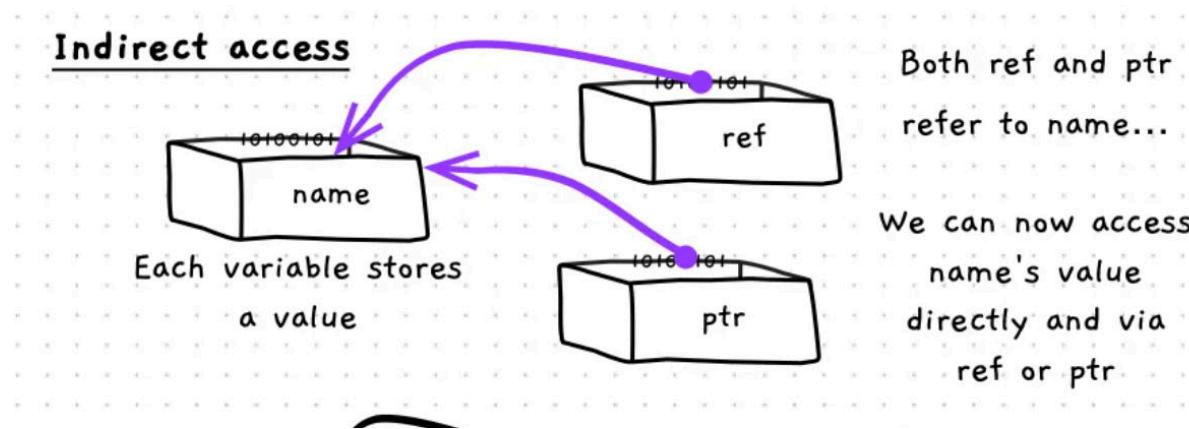
I have basically mastered the usage of loops, if statements, and switch statements in C++, as well as the usage of structures and enums. However, I haven't been exposed to knowledge about references and pointers so far.

Study to Acquire, Refresh, and/or Extend Capability

To study and master this topic I did the following:

I have started learning the knowledge of pointers and references.

First of all, here is the code explanation of references and pointers from the fieldguide.



I've learned about the concept of indirect access to variables in C++. The diagram shows that the variable `name` stores a value, and both the reference `ref` and the pointer `ptr` point to `name`. This indicates that you can get the value of `name` either by directly accessing `name` or indirectly through `ref` or `ptr`, demonstrating that references and pointers serve as ways to indirectly access variables, helping you understand how they provide different means to access and manipulate the original variable.

The screenshot shows a code editor window titled "hello.cpp". The code is as follows:

```
2
3 int main()
4 {
5     string name = "Test name";
6     string &ref = name;
7     string *ptr = &name;
8     write_line(name);
9     write_line(ref);
10    write_line(*ptr);
11 }
```

The terminal window shows the command \$./hello being run, followed by three lines of output: "Test name", "Test name", and "Test name".

```
ziyue@XMAS MINGW64 /c/users/ziyue/documents/code1/learnforc1
$ ./hello
Test name
Test name
Test name
```

I learned the basic usage of references and pointers in C++. The code defines a string variable `name`, creates a reference `ref` to it and a pointer `ptr` pointing to it, and outputs the variable's value in three ways, demonstrating that both references and pointers can access the data of the original variable.

Next is updating data using references and pointers.

```
● Hello.cpp > main()
```

```
1 #include "splashkit.h"
2
3 int main()
4 {
5     string name = "Test name";
6     string &ref = name;
7     string *ptr = &name;
8     write_line(name);
9     write_line(ref);
10    write_line(*ptr);
11
12    string other = "other";
13
14    ref = "Test update";
15    write_line(name);
16
17    *ptr = "ptr update";
18    write_line(name);|
19 }
```

```
● ziyue@XMAS MINGW64 /c/users/ziyue/documents/code1/learnforc1
$ ./hello
● Test name
Test name
Test name
Test update
ptr update
```

Complete this part of the study.

```
1 #include "splashkit.h"
2
3 int main()
4 {
5     string name = "Test name";
6     string &ref = name;
7     string *ptr = &name;
8     write_line(name);
9     write_line(ref);
10    write_line(*ptr);
11
12    string other = "other";
13
14    ref = "Test update";
15    write_line(name);
16
17    *ptr = "ptr update";
18    write_line(name);
19
20    ptr = &other;
21    write_line(*ptr);
22
23    return 0;
24 }
```

```
ziyue@XMAS MINGW64 /c/users/ziyue/documents/code1/learnforc1
● $ ./hallo
Test name
Test name
Test name
Test update
ptr update
other
```

I've learned key characteristics of references and pointers in C++. The code demonstrates that both references and pointers can access and modify the value of the original variable. For example, modifying the reference `ref` and dereferencing the pointer `ptr` both changed the value of `name`. Additionally, pointers can be reassigned to point to different variables, as shown when `ptr` later points to `other`, whereas references cannot be rebound to another variable after initialization, highlighting the distinction between the two.

Next, following the instructions in the field guide, I began a detailed study on updating user data using references.

First, the creation of the initial structure was completed.

```
30 enum role_level{
31     NO_ACCESS,
32     USER,
33     ADMIN
34 };
35
36 struct user_data{
37     string username;
38     role_level role;
39 };
40
```

The statement for completing the structure.

```
64 ~ role_level read_role()
65  {
66      int role_input = read_integer("Enter role level(0: NONE, 1: USER, 2: ADMIN): ");
67      switch (role_input)
68      {
69          case 0:
70              return NO_ACCESS;
71          case 1:
72              return USER;
73          case 2:
74              return ADMIN;
75          default:
76              write_line("Invalid role level. Setting role to NONE.");
77              return NO_ACCESS;
78      }
79  }
80
81 ~ string to_string(role_level role)
82  {
83      switch(role)
84      {
85          case 0:
86              return "NO_ACCESS";
87          case 1:
88              return "USER";
89          case 2:
90              return "ADMIN";
91          default:
92              return "Unknown";
93      }
94  }
95
```

I completed the creation of important functions, but during the process, I overlooked several semicolons at the end of statements, which caused the final code to fail to run.

```
96 void print_user(const user_data user)
97 {
98     write_line("User Details: ");
99     write_line("user Name: " + user.username);
100    write_line("Role: " + to_string(user.role));
101 }
102
103 void update_user(user_data user)
104 {
105     bool updating = true;
106     while (updating)
107     {
108         print_user(user);
109         write_line("What would you want to update? ");
110         write_line("1: User Name");
111         write_line("2: Role");
112         write_line("0: Exit");
113
114         int choice = read_integer("Enter your choice: ");
115
116         switch(choice)
117         {
118             case 1:
119                 user.username = read_string("Enter new username: ");
120                 break;
121             case 2:
122                 user.role = read_role();
123                 break;
124             case 0:
125                 updating = false;
126                 break;
127             default:
128                 write_line("Invalid choice, please try again.");
129                 break;
130         }
131     }
}
```

Complete the writing of the main part.

```
135     int main()
136     {
137         user_data user = {
138             "Ziyue",
139             USER
140         };
141
142         print_user(user);
143         update_user(user);
144         return 0;
145     }
```

OUTPUT:

```
ziyue@XMAS MINGW64 /c/users/ziyue/documents/code1/learnforc1
$ ./hello
User Details:
user Name: Ziyue
Role: USER
User Details:
user Name: Ziyue
Role: USER
What would you want to update?
1: User Name
2: Role
0: Exit
Enter your choice:
1
Enter new username:
ziyue
User Details:
user Name: ziyue
Role: USER
What would you want to update?
1: User Name
2: Role
0: Exit
Enter your choice:
0

ziyue@XMAS MINGW64 /c/users/ziyue/documents/code1/learnforc1
$ █
```

I started coding in the "Test Your Knowledge" section.

First, the creation of the initial structure and some important functions was completed.

```
8     string read_string(string prompt)
9     {
10         write_line(prompt);
11         return read_line();
12     }
13
14     int read_integer(string prompt)
15     {
16         string user_input;
17         write_line(prompt);
18         user_input = read_string(prompt);
19         while(!is_integer(user_input))
20         {
21             write_line("please enter an integer");
22             user_input = read_string(prompt);
23         }
24         int result = stoi(user_input);
25         return result;
26     }
27
28     enum menu_option
29     {
30         PRINT_DETAILS = 1,
31         DEPOSIT,
32         WITHDRAW,
33         ADD_INTEREST,
34         QUIT
35     };
36
37     struct user_data{
38         string account_name;
39         double interest_percent;
40         double balance;
41     };

```

I have completed the construction of the structure and basic functions.

```
● 52 ~ int main()
53 {
54     write_line("Simple transaction record!");
55     user_data my_account;
56
57     write("Enter account name: ");
58     my_account.account_name = read_line();
59
60     write("Enter interest percent (+/-): ");
61     my_account.interest_percent = convert_to_double(read_line());
62
63     write("Enter initial amount: ");
64     my_account.balance = convert_to_double(read_line());
65
66
67
68 }
69
```

I have completed the initial step of prompting the user for input.

```
● 52 ~ int main()
53 {
54     write_line("Simple transaction record!");
55     user_data my_account;
56
57     write("Enter account name: ");
58     my_account.account_name = read_line();
59
60     write("Enter interest percent (+/-): ");
61     my_account.interest_percent = convert_to_double(read_line());
62
63     write("Enter initial amount: ");
64     my_account.balance = convert_to_double(read_line());
65
66
67
68 }
69
```

I have completed all the functions that are selected within the switch statement.

```
43 void print_details(const user_data &acc)
44 {
45     write_line("");
46     write_line("Account: " + acc.account_name);
47     write_line("Rate: " + to_string(acc.interest_percent));
48     write_line("Balance: " + to_string(acc.balance));
49 }
50
51 void deposit(user_data &acc, double amount)
52 {
53     acc.balance += amount;
54     write_line("Balance: " + to_string(acc.balance));
55 }
56
57 void withdraw(user_data &acc, double amount)
58 {
59     acc.balance -= amount;
60     write_line("Balance: " + to_string(acc.balance));
61 }
62
63 void add_interest(user_data &acc, int days)
64 {
65     days = read_integer("Interest for how many days: ");
66     double daily_rate = acc.interest_percent / 365.0;
67
68     // Calculate interest percentage for the period
69     double interest_percentage = daily_rate * days;
70
71     // Calculate interest amount
72     double interest_amount = (interest_percentage / 100.0) * acc.balance;
73
74     // Add interest to balance
75     acc.balance += interest_amount;
76
77     write_line("Rate is " + to_string(acc.interest_percent) + "% PA = " + to_string(daily_rate * days) + "%");
78     write_line("Interest = " + to_string(interest_amount));
79 }
80 |
```

I have completed the coding process of main function.

```
71 ~ int main()
72 {
73     write_line("Simple transaction record!");
74     user_data my_account;
75
76     write("Enter account name: ");
77     my_account.account_name = read_line();
78
79     write("Enter interest percent (+/-): ");
80     my_account.interest_percent = convert_to_double(read_line());
81
82     write("Enter initial amount: ");
83     my_account.balance = convert_to_double(read_line());
84
85     bool running = true;
86 ~ while (running)
87 {
88     print_details(my_account);
89     write_line("\nActions:");
90     write_line("1: Print account details");
91     write_line("2: Deposit");
92     write_line("3: Withdraw");
93     write_line("4: Add interest");
94     write_line("5: Quit");
```

```
96     int choice = read_integer("please enter your choice: ");
97
98     switch(choice)
99     {
100         case 1:
101             print_details(my_account);
102             break;
103
104         case 2:
105             deposit(my_account, my_account.balance);
106             break;
107
108         case 3:
109             withdraw(my_account, my_account.balance);
110             break;
111
112         case 4:
113             int days = read_integer("Interest for how many days: ");
114             add_interest(my_account, days);
115
116         case 5:
117             running = false;
118             break;
119
120         default:
121             write_line("Invalid choice, please try again.");
122             break;
123     }
124 }
125 }
```

Because in the switch section, I defined "amount" as the account balance, which made it impossible for me to connect it with "userInput". Therefore, I modified the assignment of "amount".

```

● 120     int choice = read_integer("please enter your choice: ");
121     double amount;
122     switch(choice)
123     {
124         case 1:
125             print_details(my_account);
126             break;
127
128         case 2:
129             deposit(my_account, amount);
130             break;
131
132         case 3:
133             withdraw(my_account, amount);
134             break;
135
136         case 4:
137             int days;
138             add_interest(my_account, days);
139             break;
140
141         case 5:
142             running = false;
143             break;
144
145         default:
146             write_line("Invalid choice, please try again.");
147             break;
148     }
149 }
150 }
```

And I modified some statements in the deposit and withdrawal functions.

```

62     void deposit(user_data &acc, double amount)
63     {
64         amount = read_positive("Amount deposited: ");
65         acc.balance += amount;
66         write_line("Balance: " + to_string(acc.balance));
67     }
68
69     void withdraw(user_data &acc, double amount)
70     {
71         amount = read_positive("Amount withdraw: ");
72         acc.balance -= amount;
73         write_line("Balance: " + to_string(acc.balance));
74     }
75 }
```

I optimized the switch statement in the main function by using enum to map menu options. This approach significantly improves code readability - using meaningful names like `PRINT_DETAILS` and `DEPOSIT` instead of magic numbers 1, 2, 3 makes the code's intent clearer. It also enhances maintainability - when new menu options need to be added, we simply add new values to the enum rather than searching and replacing magic numbers throughout the code.

```
● 167 // Enum for menu choices
168 enum menu_option
169 {
170     PRINT_DETAILS = 1,
171     DEPOSIT,
172     WITHDRAW,
173     ADD_INTEREST,
174     QUIT
175 };
```

```
308 // Convert choice to enum
309 menu_option option = static_cast<menu_option>(choice);
310
311 switch (option)
312 {
313     case PRINT_DETAILS:
314         print_details(my_account);
315         break;
316
317     case DEPOSIT:
318         deposit(my_account, amount);
319         break;
320
321     case WITHDRAW:
322         withdraw(my_account, amount);
323         break;
324
325     case ADD_INTEREST:
326         add_interest(my_account, days);
327         break;
328
329     case QUIT:
330         running = false;
331         break;
332
333     default:
334         write_line("Invalid choice, please try again.");
335         break;
336 }
337 }
```

OUTPUT::

```
ziyue@XMAS MINGW64 /c/users/ziyue/documents/code1/banksystem
$ ./hello
● Simple transaction record!
Enter account name: Ziyue
Enter interest percent (+/-): 5.1
Enter initial amount: 1000

Account: Ziyue
Rate: 5.100000
Balance: 1000.000000

Actions:
1: Print account details
2: Deposit
3: Withdraw
4: Add interest
5: Quit
please enter your choice:
please enter your choice:
1

Account: Ziyue
Rate: 5.100000
Balance: 1000.000000

Account: Ziyue
Rate: 5.100000
Balance: 1000.000000

Actions:
1: Print account details
2: Deposit
3: Withdraw
4: Add interest
5: Quit
please enter your choice:
please enter your choice:
2
Amount deposited:
Amount deposited:
10
Balance: 1010.000000
```

```
Actions:  
1: Print account details  
2: Deposit  
3: Withdraw  
4: Add interest  
5: Quit  
please enter your choice:  
please enter your choice:  
3  
Amount withdraw:  
Amount withdraw:  
20  
Balance: 990.000000
```

```
Account: Ziyue  
Rate: 5.100000  
Balance: 990.000000
```

```
Actions:  
1: Print account details  
2: Deposit  
3: Withdraw  
4: Add interest  
5: Quit  
please enter your choice:  
please enter your choice:  
4  
Interest for how many days:  
Interest for how many days:  
30  
Rate is 5.100000% PA = 0.419178%  
Interest = 4.149863
```

```
Account: Ziyue  
Rate: 5.100000  
Balance: 994.149863
```

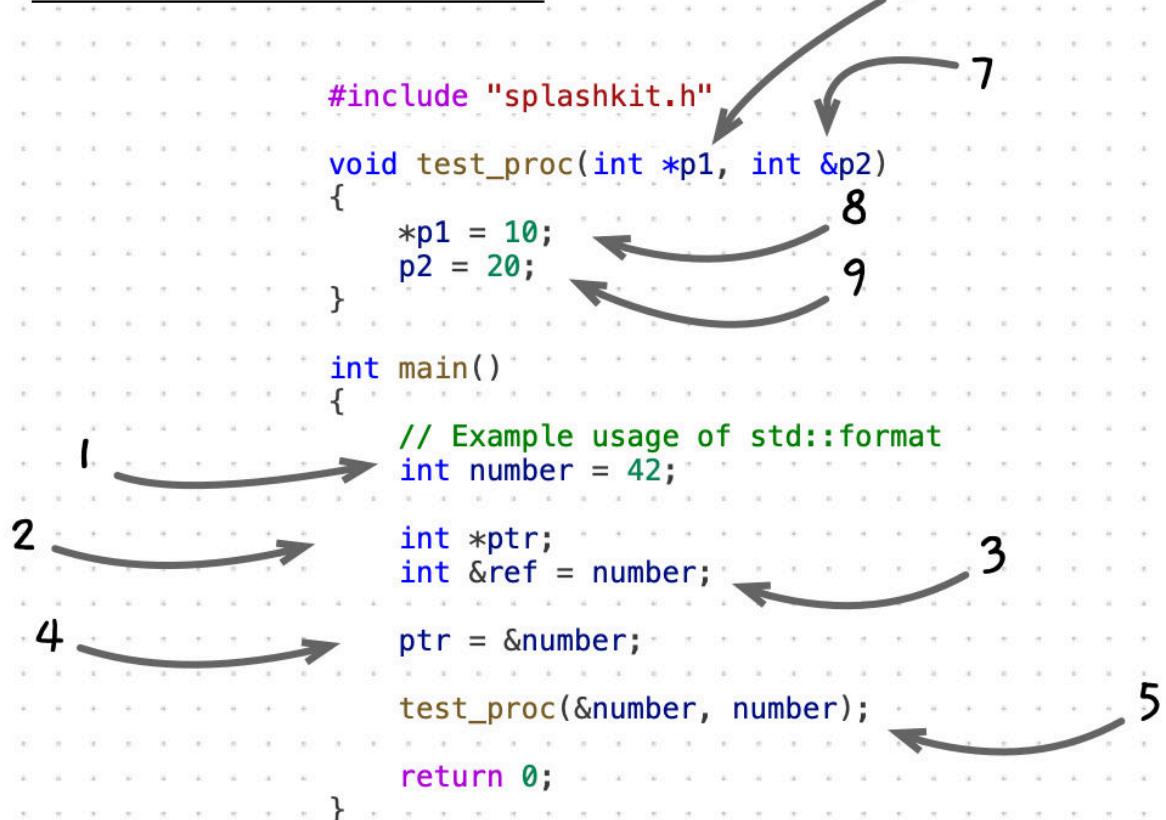
```
Actions:  
1: Print account details  
2: Deposit  
3: Withdraw  
4: Add interest  
5: Quit
```

SUCCESSFUL!!!!!!

Brief Summary of Concepts

Concept / Commands	Key Idea / Concept
Pointer	A variable that stores the memory address of another variable.
Reference	An alias that provides an alternative name for an existing variable.
Pass by value	A method where a copy of the argument's value is passed to the function.
Pass by reference	A method where the function receives a reference to the original argument, allowing modification.

Pointer and Reference



	Response
1: Describe number.	number is an integer variable initialized with the value 42.
2: Describe ptr.	ptr is a pointer to an integer, assigned to store the address of number

3: Explain what this does	it declares ref as a reference to number
4: Explain what this does.	it assigns the address of number to the pointer ptr
5: What occurs here?	it calls test_proc passing the address of number and the value of number
6: Describe p1	p1 is a pointer parameter that receives the address of number
7: Describe p2	p2 is a reference parameter that becomes an alias for a number.
8: Explain what happens here - in relation to the call indicated by arrow 5.	it dereferences p1 and changes the value of number to 10.
9: Explain what happens here - in relation to the call indicated by arrow 5.	it modifies the number directly through the reference, setting it to 20.

Describe swap

The screenshot shows a Visual Studio Code interface with the following details:

- EXPLORER**: Shows the project structure under **C1WORK**, including files `hello.cpp`, `hello.exe`, `utilities.cpp`, and `utilities.h`.
- OPEN EDITORS**: Shows the current editor has `hello.cpp` open.
- TERMINAL**: Shows the command line output:

```
ziyue@XMAS MINGW64 /c/users/ziyue/documents/code1/c1work
$ clang++ hello.cpp -l splashkit -o hello

ziyue@XMAS MINGW64 /c/users/ziyue/documents/code1/c1work
$ ./hello
Before test_proc:
number = 42
After test_proc:
number = 20
```
- EDITOR**: Shows the content of `hello.cpp` with syntax highlighting for C++ code.

```
hello.cpp x

G: hello.cpp > main()
1 #include <cstdio>
2
3 void test_proc(int *p1, int &p2)
4 {
5     *p1 = 10;
6     p2 = 20;
7 }
8
9 int main()
10 {
11     int number = 42;
12     int *ptr;
13     int &ref = number;
14
15     ptr = &number;
16
17     printf("Before test_proc:\n");
18     printf("number = %d\n", number);
19
20     test_proc(&number, number);
21
22     printf("After test_proc:\n");
23     printf("number = %d\n", number);
24
25
26 }
```

The function demonstrates that both pointers (using `*`) and references can modify the original variable. However, in this specific case, the reference assignment `p2 = 20` happens last, so the final value of the number becomes 20, overwriting the previous assignment of 10.

Reflection

What is the most important thing you learned from this and why?

The most important thing learned is understanding how references and pointers enable functions to modify original variables, enhancing code efficiency and flexibility.

How do you see references helping you modularise larger programs?

References help modularize larger programs by allowing functions to safely and clearly manipulate data without copying, improving interface design and reducing errors.

Many modern languages make extensive use of references, but do not include an equivalent of pointers. Why do you think this may be?

Modern languages omit pointers to simplify memory management, reduce complexity, and prevent common errors like dangling pointers or memory leaks, relying instead on safer reference mechanisms.

References

[Generative AI acknowledgement](#)

No GenAI used in this work.

This file has additional line breaks applied by OnTrack because they contain lines longer than the configured limit. Lines over 1000 characters long have been truncated to limit PDF page count. The orginal submission can be retrieved via the "Download Uploaded Files" function.

```
1 #include "splashkit.h"
2 #include "utilities.h"
3
4 using std::stoi;
5 using std::to_string;
6
7 // Enum for menu choices
8 enum menu_option
9 {
10     PRINT_DETAILS = 1,
11     DEPOSIT,
12     WITHDRAW,
13     ADD_INTEREST,
14     QUIT
15 };
16
17 // Function to display menu options
18 void print_menu()
19 {
20     write_line("\nActions:");
21     write_line("1: Print account details");
22     write_line("2: Deposit");
23     write_line("3: Withdraw");
24     write_line("4: Add interest");
25     write_line("5: Quit");
26 }
27
28 // Function to read a string input from user with prompt
29 string read_string(string prompt)
30 {
31     write_line(prompt);
32     return read_line();
33 }
34
35 // Function to read and validate integer input from user
36 int read_integer(string prompt)
37 {
38     string user_input;
39     write_line(prompt);
40     user_input = read_string(prompt);
41     // Keep asking until valid integer is entered
42     while (!is_integer(user_input))
43     {
44         write_line("please enter an integer");
45         user_input = read_string(prompt);
46     }
47     int result = stoi(user_input);
48     return result;
49 }
```

```
51 // Function to read and validate positive double input
52 int read_positive(string prompt)
53 {
54     string user_input;
55     write_line(prompt);
56     user_input = read_string(prompt);
57     while (!is_integer(user_input))
58     {
59         write_line("please enter an integer");
60         user_input = read_string(prompt);
61     }
62     int result = stoi(user_input);
63     while (result < 0)
64     {
65         write_line("Please enter a positive integer: ");
66         result = stoi(read_line());
67     }
68     return result;
69 }
70
71 // Structure to store bank account information
72 struct user_data
73 {
74     string account_name;
75     double interest_percent;
76     double balance;
77 };
78
79 // Function to display account details
80 void print_details(const user_data &acc)
81 {
82     write_line("");
83     write_line("Account: " + acc.account_name);
84     write_line("Rate: " + to_string(acc.interest_percent));
85     write_line("Balance: " + to_string(acc.balance));
86 }
87
88 // Function to deposit money into account
89 void deposit(user_data &acc, double amount)
90 {
91     amount = read_positive("Amount deposited: ");
92     // Get valid deposit amount
93     acc.balance += amount; // Add amount to balance
94     write_line("Balance: " + to_string(acc.balance)); // Display updated balance
95 }
96
97 // Function to withdraw money from account
98 void withdraw(user_data &acc, double amount)
99 {
100    amount = read_positive("Amount withdrawn: ");
101   acc.balance -= amount; // Subtract amount from balance
102   write_line("Balance: " + to_string(acc.balance));
103 }
```

```
105 void add_interest(user_data &acc, int days)
106 {
107     days = read_integer("Interest for how many days: ");
108     double daily_rate = acc.interest_percent / 365.0;
109
110     // Calculate interest percentage for the period
111     double interest_percentage = daily_rate * days;
112
113     // Calculate interest amount
114     double interest_amount = (interest_percentage / 100.0) * acc.balance;
115
116     // Add interest to balance
117     acc.balance += interest_amount;
118
119     write_line("Rate is " + to_string(acc.interest_percent) + "% PA = " +
120     ↵ to_string(daily_rate * days) + "%");
121     write_line("Interest = " + to_string(interest_amount));
122 }
123
124 // Main function with enum-based switch statement
125 int main()
126 {
127     // write_line("Simple transaction record!");
128     user_data my_account; // Create account instance
129     // Get account information from user
130     write("Enter account name: ");
131     my_account.account_name = read_line();
132
133     write("Enter interest percent (+/-): ");
134     my_account.interest_percent = convert_to_double(read_line());
135
136     write("Enter initial amount: ");
137     my_account.balance = convert_to_double(read_line());
138
139     bool running = true; // Control variable for main loop
140     while (running)
141     {
142         print_details(my_account);
143         print_menu();
144
145         int choice = read_integer("please enter your choice: ");
146         double amount;
147         int days;
148
149         // Convert choice to enum
150         menu_option option = static_cast<menu_option>(choice);
151
152         switch (option)
153         {
154             case PRINT_DETAILS:
155                 print_details(my_account);
156                 break;
157
158             case DEPOSIT:
```

```
158         deposit(my_account, amount);
159         break;
160
161     case WITHDRAW:
162         withdraw(my_account, amount);
163         break;
164
165     case ADD_INTEREST:
166         add_interest(my_account, days);
167         break;
168
169     case QUIT:
170         running = false;
171         break;
172
173     default:
174         write_line("Invalid choice, please try again.");
175         break;
176     }
177 }
178
179 return 0;
180 }
```

12 Structuring Code

As the logic in your program grows, you will need better ways to organise your code. Learn how to structure your code to better organise the logic and your thoughts using functions and procedures.

Outcome	Description		
Date	Author	Comment	
TLO1		Design and construct programs that are logically organised in	
TLO2		Use parameters and local variables effectively to control data	
TLO3		Apply good programming practices when structuring code, in	
TLO4		Document prior knowledge, study steps, concept summaries, i	
2025/08/06 20:23	Ziyue Meng	Planned date adjusted to 22 Aug.	
2025/08/17 18:14	Ziyue Meng	Working On It	
2025/08/19 19:30	Ziyue Meng	Ready for Feedback	
2025/09/01 11:18	Ganesh Krishnasamy	Hi Ziyue.	
2025/09/01 11:19	Ganesh Krishnasamy	Your code would benefit from more inline comments explaining your logic and purpose. Please update your documentation.	
2025/09/01 11:23	Ganesh Krishnasamy	Also, please include white spaces between the main block codes.	
2025/09/01 11:23	Ganesh Krishnasamy	Fix and Resubmit	
2025/09/01 14:15	Ziyue Meng	Ready for Feedback	
2025/09/01 14:16	Ziyue Meng	I have modified my work and reupload it, thank you for your guidance!	
2025/09/05 20:15	Ganesh Krishnasamy	Also, remove unwanted codes from line 161 to 235.	
2025/09/05 20:15	Ganesh Krishnasamy	Fix and Resubmit	
2025/09/05 20:24	Ganesh Krishnasamy	Also, I can't see any engagement with the guided tour and concepts for Structuring Code. At least, engagement with one of the guided tour and capture the learning journey.	
2025/09/05 20:24	Ganesh Krishnasamy	Fix and Resubmit	
2025/09/06 12:39	Ziyue Meng	Ready for Feedback	
2025/09/06 12:39	Ziyue Meng	I have modified my work and reupload it, thank you for your guidance!	
2025/09/09 21:04	Ganesh Krishnasamy	You can further modularised your code especially in your main () program.	
2025/09/09 21:04	Ganesh Krishnasamy	Fix and Resubmit	
2025/09/10 11:43	Ziyue Meng	Ready for Feedback	
2025/09/10 11:46	Ziyue Meng	I use comments and line spacing to make my data more understandable. However, I don't use struct() and enum() because these concepts weren't covered at the P6 stage. Thank you.	
2025/09/10 21:23	Ganesh Krishnasamy	Complete	

INTRODUCTION TO PROGRAMMING

ONTRACK SUBMISSION

Structuring Code

Submitted By:

Ziyue MENG
zmen0034
2025/09/10 11:43

Tutor:

Ganesh KRISHNASAMY

Task Outcomes	Supports
TLO1 Design and construct programs that are logically organised into functions and procedures to improve clarity and reuse.	ULO2 Programs, Functions, and Types
TLO2 Use parameters and local variables effectively to control data flow and manage program state without relying on shared state.	ULO2 Programs, Functions, and Types ULO4 Use good programming practices
TLO3 Apply good programming practices when structuring code, including clear naming, separation of concerns, and avoidance of poor practices such as global variables.	ULO4 Use good programming practices
TLO4 Document prior knowledge, study steps, concept summaries, and reflections to explain how structuring code supports program development.	ULO4 Use good programming practices

September 10, 2025



Structuring Code

Name: Ziyue Meng
Student ID: 36035432
Date: 19/8/2025

Learning Journey and Evidence

Prior Knowledge and Experience

I have not only grasped the core differences between C++ and C#, but also gained an in-depth understanding of the distinctions in their syntax structures and application scenarios. For example, C++ is closer to the low-level, allowing direct memory manipulation, and is suitable for system-level development and high-performance applications; while C#, relying on the .NET framework, has more advantages in terms of security and development efficiency.

Meanwhile, I have also proficiently mastered the usage of loop structures (such as for, while, do-while) and if conditional statements in C++. Through extensive practice, I can flexibly choose the appropriate loop type to handle repetitive tasks according to specific needs, and can also use if-else nested structures to accurately implement complex conditional judgment logic, which has laid a solid foundation for me to write efficient and clear C++ programs.

Study to Acquire, Refresh, and/or Extend Capability

While working on this assignment, I received a message from the professor. He helped me fix those errors in the code that don't affect the operation. However, since I had already completed part of the P6 study, such errors will still appear in the first half of this file, but I have resolved them in the second half. Please forgive me for this.



Charlotte Pierce STAFF

1小时前



Try this!



1. Run `gcc -v -E -x c++ -` in a terminal
2. In the output, find the line `#include <...>` search starts here:
3. Copy every path under that, until you get to `End of search list`.
4. Paste those paths into the VS code settings file, in the `"C_Cpp.default.systemIncludePath":` section. Add quotes around each path, and a comma at the end of each line to format the JSON correctly

This should sort out VS Code struggling to find the libraries. Let a TA know in class if you need help with this!

评论 ...

M ~

```
zhiyue@XMAS MINGW64 ~
$ gcc -v -E -x c++ -
Using built-in specs.
COLLECT_GCC=C:\msys64\mingw64\bin\gcc.exe
Target: x86_64-w64-mingw32
Configured with: ../gcc-15.1.0/configure --prefix=/mingw64 --with-local-prefix=/mingw64/local --with-native-system-header-dir=/mingw64/include --libexecdir=/mingw64/lib --enable-bootstrap --enable-checking=release --with-arch=nocona --with-tune=generic --enable-mingw-wildcard --enable-languages=c,c++,fortran,ada,objc,obj-c++,objc++,jit --enable-shared --enable-static --enable-libatomic --enable-threads=posix --enable-graphite --enable-fuzzy-dynamic-string --enable-libstdcxx-backtrace=yes --enable-libstdcxx-filesystem-ts --enable-libstdcxx-time --disable-libstdcxx-pch --enable-lto --enable-libgomp --disable-libssp --disable-multilib --disable-rpath --disable-win32-registry --disable-nls --disable-error --disable-symvers --with-libiconv --with-system-zlib --with-gmp=/mingw64 --with-mpfr=/mingw64 --with-mpc=/mingw64 --with-isl=/mingw64 --with-pkgversion='Rev8, Built by MSYS2 project' --with-bugurl=https://github.com/msys2/MINGW-packages/issues --with-gnu-as --with-gnu-ld --with-libstdcxx-zeninfo=yes --disable-libstdcxx-debug --enable-plugin --with-boot-ldflags=-static-libstdc++ --with-stage1-ldflags=-static-libstdc++
Thread model: posix
Supported LTO compression algorithms: zlib zstd
gcc version 15.1.0 (Rev8, Built by MSYS2 project)
COLLECT_GCC_OPTIONS='-v' '-E' '-mtune=generic' '-march=nocona'
C:/msys64/mingw64/bin/../lib/gcc/x86_64-w64-mingw32/15.1.0/cc1plus.exe -E -quiet -v -iprefix C:/msys64/mingw64/bin/../lib/gcc/x86_64-w64-mingw32/15.1.0/ -D_REENTRANT - -mtune=generic -march=nocona -dumbbase -
ignoring nonexistent directory "C:/msys64/mingw64/bin/../lib/gcc/x86_64-w64-mingw32/15.1.0/../../../../x86_64-w64-mingw32/include"
ignoring duplicate directory "C:/msys64/mingw64/lib/gcc/../../../lib/gcc/x86_64-w64-mingw32/15.1.0/../../../../include/c++/15.1.0"
ignoring duplicate directory "C:/msys64/mingw64/lib/gcc/../../../lib/gcc/x86_64-w64-mingw32/15.1.0/../../../../include/c++/15.1.0/x86_64-w64-mingw32"
ignoring duplicate directory "C:/msys64/mingw64/lib/gcc/../../../lib/gcc/x86_64-w64-mingw32/15.1.0/../../../../include/c++/15.1.0/backward"
ignoring duplicate directory "C:/msys64/mingw64/lib/gcc/../../../lib/gcc/x86_64-w64-mingw32/15.1.0/include"
ignoring nonexistent directory "D:/M/msys64/mingw64/include"
ignoring nonexistent directory "/mingw64/include"
ignoring duplicate directory "C:/msys64/mingw64/lib/gcc/../../../lib/gcc/x86_64-w64-mingw32/15.1.0/include-fixed"
ignoring nonexistent directory "C:/msys64/mingw64/lib/gcc/../../../lib/gcc/x86_64-w64-mingw32/15.1.0/../../../../x86_64-w64-mingw32/include"
ignoring nonexistent directory "D:/M/msys64/mingw64/include"
#include "..." search starts here:
#include <...> search starts here:
C:/msys64/mingw64/bin/../lib/gcc/x86_64-w64-mingw32/15.1.0/../../../../include/c++/15.1.0
C:/msys64/mingw64/bin/../lib/gcc/x86_64-w64-mingw32/15.1.0/../../../../include/c++/15.1.0/x86_64-w64-mingw32
C:/msys64/mingw64/bin/../lib/gcc/x86_64-w64-mingw32/15.1.0/../../../../include/c++/15.1.0/backward
C:/msys64/mingw64/bin/../lib/gcc/x86_64-w64-mingw32/15.1.0/include
C:/msys64/mingw64/bin/../lib/gcc/x86_64-w64-mingw32/15.1.0/../../../../include
C:/msys64/mingw64/bin/../lib/gcc/x86_64-w64-mingw32/15.1.0/include-fixed
End of search list.
```

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer**: Shows a folder structure including "RUN AND DEBUG...", "hello.cpp", "Settings", and "settings.json".
- Run and Debug**: A button for running and debugging the project.
- Terminal**: A terminal window showing the command: `zhiyue@XIAO-MINGW64 /c/users/zhiyue/documents/code1/learnforp6`.
- Code Editor**: The `settings.json` file open, displaying configuration for the terminal and C/C++ compiler.
- Bottom Bar**: Includes tabs for PROBLEMS, TERMINAL, OUTPUT, and PORTS, along with a bash terminal icon.

```
C:\>Users>zhiyue>AppData>Roaming>Code>User>{ settings.json > ...
2   "terminal.integrated.profiles.windows": {
3     "MSYS2": {
4       "args": [
5         ],
6         "env": {
7           "MSYSTEM": "MINGW64",
8           "CHERE_INVOKING": "1"
9         }
10      },
11    },
12    "terminal.integrated.defaultProfile.windows": "MSYS2",
13    "terminal.integrated.env.windows": {
14      "MSYSTEM": "MINGW64",
15      "CHERE_INVOKING": "1"
16    },
17    "C_Cpp.default.systemIncludePath": [
18      "C:\msys64\mingw64\bin",
19      "C:\msys64\mingw64\include",
20      "${default}",
21      "C:\msys64\mingw64\bin../lib/gcc/x86_64-w64-mingw32/15.1.0/../../../../include/c++/15.1.0",
22      "C:\msys64\mingw64\bin../lib/gcc/x86_64-w64-mingw32/15.1.0/../../../../include/c++/15.1.0/x86_64-w64-mingw32",
23      "C:\msys64\mingw64\bin../lib/gcc/x86_64-w64-mingw32/15.1.0/../../../../include/c++/15.1.0/backward",
24      "C:\msys64\mingw64\bin../lib/gcc/x86_64-w64-mingw32/15.1.0/include",
25      "C:\msys64\mingw64\bin../lib/gcc/x86_64-w64-mingw32/15.1.0/../../../../include",
26      "C:\msys64\mingw64\bin../lib/gcc/x86_64-w64-mingw32/15.1.0/include-fixed"
27    ],
28    "files.autoSave": "afterDelay",
29    "editor.formatOnSave": true,
30    "C_Cpp.autocomplete": "disabled"
31  },
32  "files.exclude": {
33    "**/.git": true,
34    "**/.hg": true,
35    "**/.svn": true,
36    "**/.bzr": true
37  }
38 }
39 }
40 }
41 }
42 }
43 }
44 }
45 }
46 }
```

```
..  ⌂ hello.cpp X  ⌂ Settings  {} settings.json
  ⌂ hello.cpp > ⌂ main()
43
44     #include "splashkit.h"
45     using std::stoi;
46     using std::to_string;
47
48     string read_string(string prompt)
49     {
50         write(prompt);
51         return read_line();
52     }
53
54     int read_integer(string prompt)
55     {
56         string line;
57         line = read_string(prompt);
58         while (!is_integer(prompt))
59         {
60             write("please enter a whole number: ");
61             line = read_string(prompt);
62         }
63         int result;
64         result = stoi(prompt);
65
66         return result;
67     }
68
69     bool perform_guess(int guess_number, int target)
70     {
71         int guess;
72         read_integer("Guess" + to_string(guess_number) + ":" );
73     }
74
75     int main()
76     {
```

PROBLEMS DEBUG CONSOLE TERMINAL OUTPUT PORTS

```
ziyue@XMAS MINGW64 /c/users/ziyue/documents/code1/learnforp6
$
```

To study and master this topic I did the following:

```
<18/8/2025>
ziyue@XMAS MINGW64 ~
$ cd /c/users/documents/code1
-bash: cd: /c/users/documents/code1: No such file or directory

ziyue@XMAS MINGW64 ~
$ cd /c/users/ziyue/documents/code1

ziyue@XMAS MINGW64 /c/users/ziyue/documents/code1
$ mkdir learnforp6

ziyue@XMAS MINGW64 /c/users/ziyue/documents/code1
$ cd learnforp6

ziyue@XMAS MINGW64 /c/users/ziyue/documents/code1/learnforp6
$ touch hello.cpp

ziyue@XMAS MINGW64 /c/users/ziyue/documents/code1/learnforp6
$ code .

ziyue@XMAS MINGW64 /c/users/ziyue/documents/code1/learnforp6
$ |
```

Parameters

Parameters are declared within the parenthesis after the function or procedure name.

There is one string parameter named prompt created here

```
string read_string(string prompt)
{
    string result;
    write(prompt);
    result = read_line();
    return result;
}
```

In the function or procedure's instructions, prompt is a variable - it just already has a value! No assignment statement needed. The caller provides the value in the function call.

Read Integer

Looking at other terminal input in our programs we also have places where we want to read integer values from the user. For example, the following code was in our change calculator program. We can simplify this a little by using our `read_string` function, but we could further refine this by creating a `read_integer` function that performs this validation as well.

```
write("Cost of item in cents: ");
line = read_line();
while ( ! is_integer(line) )
{
    write_line("Please enter a whole number.");
    write("Cost of item in cents: ");
    line = read_line();
}
int cost_of_item = stoi(line);
```

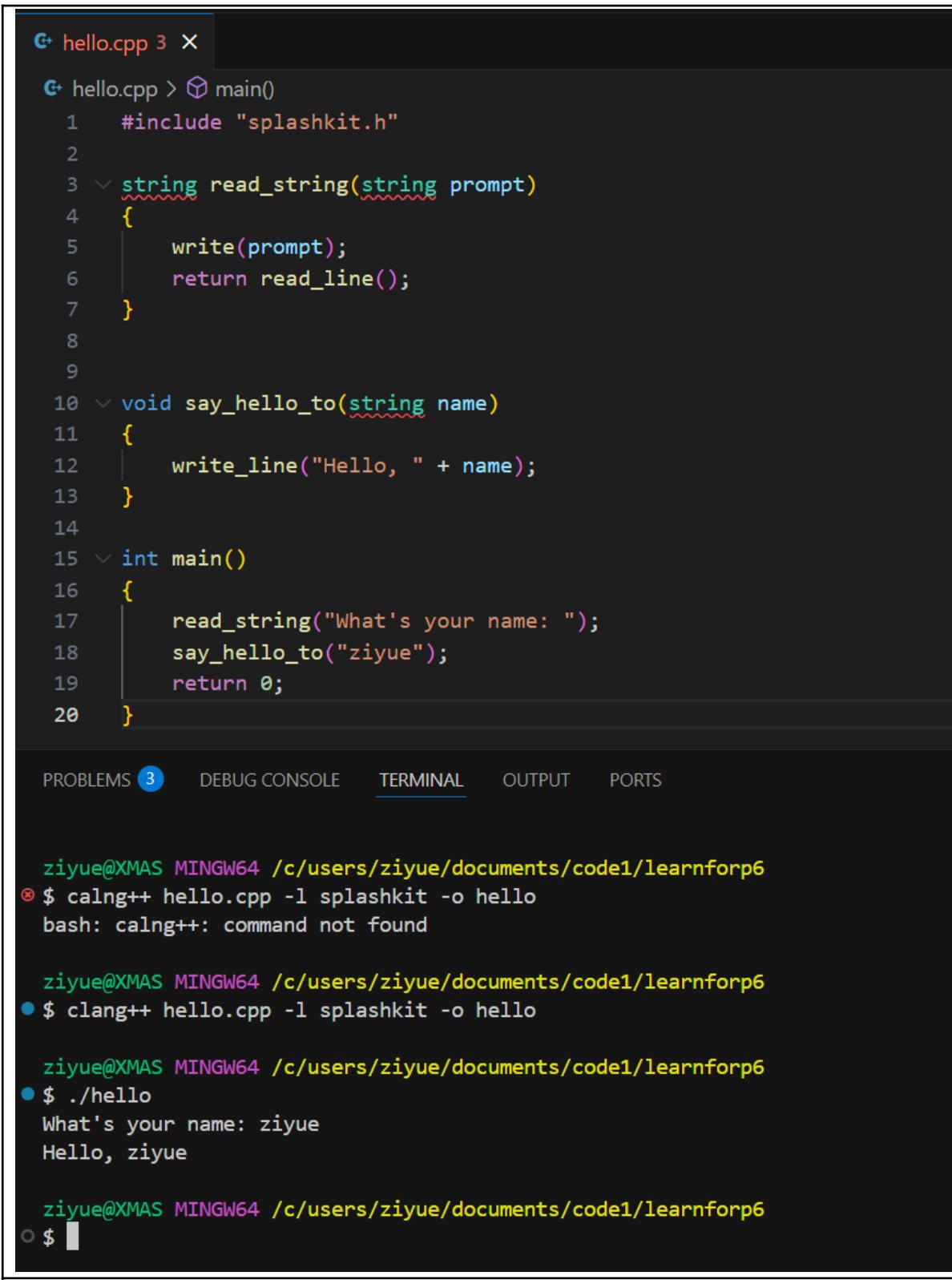
```
hello.cpp 1 ×
hello.cpp > ⚙ say_hello_to(string)
1 #include "splashkit.h"
2
3 void say_hello_to(string name)
4 {
5     write_line("Hello, " + name);
6 }
7
8 int main()
9 {
10    say_hello_to("ziyue");
11    return 0;
12 }
```

PROBLEMS 1 DEBUG CONSOLE TERMINAL OUTPUT PORTS

```
ziyue@XMAS MINGW64 /c/users/ziyue/documents/code1/learnforp6
● $ clang++ hello.cpp -l splashkit -o hello

ziyue@XMAS MINGW64 /c/users/ziyue/documents/code1/learnforp6
● $ ./hello
Hello, ziyue

ziyue@XMAS MINGW64 /c/users/ziyue/documents/code1/learnforp6
○ $ █
```



```
hello.cpp 3
hello.cpp > ↗ main()
1 #include "splashkit.h"
2
3 ↘ string read_string(string prompt)
4 {
5     write(prompt);
6     return read_line();
7 }
8
9
10 ↘ void say_hello_to(string name)
11 {
12     write_line("Hello, " + name);
13 }
14
15 ↘ int main()
16 {
17     read_string("What's your name: ");
18     say_hello_to("ziyue");
19     return 0;
20 }
```

PROBLEMS 3 DEBUG CONSOLE TERMINAL OUTPUT PORTS

```
ziyue@XMAS MINGW64 /c/users/ziyue/documents/code1/learnforp6
$ clang++ hello.cpp -l splashkit -o hello
bash: clang++: command not found

ziyue@XMAS MINGW64 /c/users/ziyue/documents/code1/learnforp6
$ clang++ hello.cpp -l splashkit -o hello

ziyue@XMAS MINGW64 /c/users/ziyue/documents/code1/learnforp6
$ ./hello
What's your name: ziyue
Hello, ziyue

ziyue@XMAS MINGW64 /c/users/ziyue/documents/code1/learnforp6
$
```

These are two functions used to read strings and integers respectively, and they are also the most basic functions in C++. I have completed the learning of these two functions by following the Guide Tour.

I have learned about the function in C++.

```
hello.cpp 9 ×
hello.cpp > main()
1 #include "splashkit.h"
2 using std::stoi;
3 using std::to_string;
4
5
6 string read_string(string prompt)
7 {
8     write(prompt);
9     return read_line();
10}
11
12
13 int read_integer(string prompt)
14 {
15     string line;
16     line = read_string(prompt);
17     while(! is_integer(line))
18     {
19         write("Please enter a number: ");
20         line = read_string(prompt);
21     }
22     int result;
23     result = stoi(line);
24     return result;
25}
26
27
28 void say_hello_to(string name)
29 {
30     write_line("Hello, " + name);
31 }
32
33
34 int main()
35 {
36     read_string("What's your name: ");
37     say_hello_to("ziyue");
38
39     int number;
40     number = read_integer("How old are you: ");
41     write_line("You are " + to_string(number) + "!");
42
43     return 0;
44}
```

```
ziyue@XMAS MINGW64 /c/users/ziyue/documents/code1/learnforp6
● $ clang++ hello.cpp -l splashkit -o hello

ziyue@XMAS MINGW64 /c/users/ziyue/documents/code1/learnforp6
● $ ./hello
What's your name: ziyue
Hello, ziyue
How old are you: 18
You are 18!

ziyue@XMAS MINGW64 /c/users/ziyue/documents/code1/learnforp6
○ $ █
```

I started on the main work.

```
ziyue@XMAS MINGW64 /c/users/ziyue/documents/code1
$ mkdir AFL_Score

ziyue@XMAS MINGW64 /c/users/ziyue/documents/code1
$ cd AFL_Score

ziyue@XMAS MINGW64 /c/users/ziyue/documents/code1/AFL_Score
$ touch hello.cpp

ziyue@XMAS MINGW64 /c/users/ziyue/documents/code1/AFL_Score
$ code .

ziyue@XMAS MINGW64 /c/users/ziyue/documents/code1/AFL_Score
$
```

First, I created several important functions.

```

1  #include "splashkit.h"
2  using std::stoi;
3  using std::to_string;
4
5
6  string read_string(string prompt)
7  {
8      write(prompt);
9      return read_line();
10 }
11
12 int read_integer(string prompt)
13 {
14     string line;
15     line = read_string(prompt);
16     while(! is_integer(line))
17     {
18         write("Please enter a integer: ");
19         line = read_string(prompt);
20     }
21     int result = stoi(line);
22     return result;
23 }
24
25 int calculator(int goal, int behinds)
26 {
27     int score;
28     score = goal * 6 + behinds;
29     return score;
30 }
31

```

Then, I wrote the basic framework.

```

int main()
{
    write_line("Welcome to the AFL score calculator!");
    write_line("Enter the team 1 details: ");
    string name1 = read_string("name: ");
    int goal1 = read_integer("goals: ");
    int behind1 = read_integer("behinds: ");
    int score1 = calculator(goal1, behind1);
    write_line(name1 + ": " + to_string(goal1) + ", " + to_string(behind1) + ", " + to_string(score1));

    string name2 = read_string("name: ");
    int goal2 = read_integer("goals: ");
    int behind2 = read_integer("behinds: ");
    int score2 = calculator(goal2, behind2);
    write_line(name2 + ": " + to_string(goal2) + ", " + to_string(behind2) + ", " + to_string(score2));
}

```

Then, I wrote the basic framework.

```
34 ~ void compare(int x, int y)
35 {
36     if(x > y)
37     {
38         write_line(to_string(x) + ">" + to_string(y) + "The" + " wins!");
39     }
40     else if(x < y)
41     {
42         write_line(to_string(y) + ">" + to_string(x) + "The" + " wins!");
43     }
44     else
45     {
46         write_line(to_string(y) + "=" + to_string(x) + "Draw!");
47     }
48
49     return;
50 }
```

When I wanted to create a function for the step of comparing sizes, I found that even if the size comparison was completed, I couldn't map the numbers to the team names.

I realized that in C++, I hadn't learned a statement that can achieve a one-to-one correspondence like the dictionary function in Python.

So, I gave up naming this part as a function.

I had to write the process of comparing sizes in the main function.

```
if(score1 > score2)
{
    write_line(to_string(score1) + ">" + to_string(score2) + "\nThe " + name1 + " wins!");
}
else if(score1 < score2)
{
    write_line(to_string(score2) + ">" + to_string(score1) + "\nThe " + name2 + " wins!");
}
else
{
    write_line(to_string(score1) + "=" + to_string(score2) + "\nDraw!");
```

I completed the step of prompting for number selection.

```
60     string quit;
61     string number;
62     int option;
63     while(true)
64     {
65         write_line("Menu:");
66         write_line("1: Update " + name1 + " goals");
67         write_line("2: Update " + name1 + " behinds");
68         write_line("3: Update " + name2 + " goals");
69         write_line("4: Update " + name2 + " behinds");
70         write_line("5: Print details");
71         write_line("6: Quit");
72
73         number = read_line();
74         option = stoi(number);
75         while(true)
76         {
77             if(option>=1 && option<=6)
78             {
79                 break;
80             }
81             else
82             {
83                 write_line("Please enter number between 1 and 6: ");
84                 option = stoi(read_line());
85             }
86         }
87     }
88 }
```

In the subsequent selection process, my first thought was to use the case statement to accomplish it.

But when running, I found that when I selected 1/2/3/4, 5 and 6 would automatically run next, and I didn't know why.

```

146    switch(option)
147    {
148        case 1:
149            write_line("goals: ");
150            goal1 = stoi(read_line());
151        case 2:
152            write_line("behinds: ");
153            behind1 = stoi(read_line());
154        case 3:
155            write_line("goals: ");
156            goal2 = stoi(read_line());
157        case 4:
158            write_line("behinds: ");
159            behind2 = stoi(read_line());
160    case 5:{
161        int score1 = calculator(goal1, behind1);
162        int score2 = calculator(goal2, behind2);
163        if(score1 > score2)
164        {
165            write_line(to_string(score1) + ">" + to_string(score2) + "The" + name1 + "wins!");
166        }
167        else if(score1 < score2)
168        {
169            write_line(to_string(score2) + ">" + to_string(score1) + "The" + name2 + "wins!");
170        }
171        else
172        {
173            write_line(to_string(score1) + "=" + to_string(score2) + "Draw!");
174        }
175    }

```

```

176    case 6:
177        write_line("Are you sure you want to quit? [Y/n]:");
178        quit = read_line();
179        while(true)
180        {
181            if(quit == "Y")
182            {
183                write_line("Bye!");
184                return 0;
185            }
186            else if(quit == "n")
187            {
188                break;
189            }
190            else
191            {
192                write_line("Please enter Y/n");
193                quit = read_line();
194            }
195        }
196    }

```

So, I changed this section of code to the if-else if mode.

```

87    if(option == 1)
88    {
89        write_line("goals: ");
90        goal1 = stoi(read_line());
91    }
92    else if(option == 2)
93    {
94        write_line("behinds: ");
95        behind1 = stoi(read_line());
96    }
97    else if(option == 3)
98    {
99        write_line("goals: ");
100       goal2 = stoi(read_line());
101    }
102    else if(option == 4)
103    {
104        write_line("behinds: ");
105        behind2 = stoi(read_line());
106    }

```

```

107    else if(option == 5)
108    {
109        int score1 = calculator(goal1, behind1);
110        int score2 = calculator(goal2, behind2);
111        if(score1 > score2)
112        {
113            write_line(to_string(score1) + ">" + to_string(score2) + "\nThe " + name1 + " wins!");
114        }
115        else if(score1 < score2)
116        {
117            write_line(to_string(score2) + ">" + to_string(score1) + "\nThe " + name2 + " wins!");
118        }
119        else
120        {
121            write_line(to_string(score1) + "=" + to_string(score2) + "\nDraw!");
122        }
123    }
124    else if(option == 6)
125    {
126        write_line("Are you sure you want to quit? [Y/n]:");
127        quit = read_line();
128        while(true)
129        {
130            if(quit == "Y")
131            {
132                write_line("Bye!");
133                return 0;
134            }
135            else if(quit == "n")
136            {
137                break;
138            }
139            else
140            {
141                write_line("Please enter Y/n");
142                quit = read_line();
143            }

```

And I commented out the case part (this step can be seen in the subsequent code part).

```
196         // }
197     }
198
199     return 0;
200 }
```

Completed and ran it.

```
ziyue@XMAS MINGW64 /c/users/ziyue/documents/code1/AFL_Score
● $ clang++ hello.cpp -l splashkit -o hello

ziyue@XMAS MINGW64 /c/users/ziyue/documents/code1/AFL_Score
● $ ./hello
Welcome to the AFL score calculator!
Enter the team 1 details:
name: Hawks
goals: five
Please enter a integer: goals: 5
behinds: 12.1
Please enter a integer: behinds: 12
Hawks: 5, 12, 42
name: Cats
goals: 7
behinds: 3
Cats: 7, 3, 45
45>42
The Cats wins!
Menu:
1: Update Hawks goals
2: Update Hawks behinds
3: Update Cats goals
4: Update Cats behinds
5: Print details
6: Quit
7
Please enter number between 1 and 6:
4
behinds:
0
Menu:
1: Update Hawks goals
2: Update Hawks behinds
3: Update Cats goals
4: Update Cats behinds
5: Print details
6: Quit
5
42=42
Draw!
Menu:
1: Update Hawks goals
2: Update Hawks behinds
3: Update Cats goals
4: Update Cats behinds
5: Print details
```

```

Menu:
1: Update Hawksgoals
2: Update Hawks behinds
3: Update Cats goals
4: Update Cats behinds
5: Print details
6: Quit
6
Are you sure you want to quit? [Y/n]:
1
Please enter Y/n
n
Menu:
1: Update Hawksgoals
2: Update Hawks behinds
3: Update Cats goals
4: Update Cats behinds
5: Print details
6: Quit
6
Are you sure you want to quit? [Y/n]:
Y
Bye!

ziyue@XMAS MINGW64 /c/users/ziyue/documents/code1/AFL_Score
○ $ █

```

SUCCESSFUL!!!!!!!

Brief Summary of Concepts

Concept	Key Idea / Concept
Procedure	A sequence of ordered steps carried out to complete a specific task, usually without returning a value, focusing on performing actions.
Function	A reusable block of code that can accept inputs (parameters) and return a value, used for modular - programming.
Return	Enables a function to send the calculation result or a specific value back to the place where it was called, used to end the function execution and pass data.
Local Variable	A variable declared inside a function or code block. Its scope is limited to that function or code block, and it is usually destroyed when the function execution ends.

Parameter	A variable specified in a function definition, used to pass data into the function when the function is called, and it is the way for a function to receive inputs.
Global Variable	A variable declared outside of functions, which can be accessed and modified in multiple parts (different functions, etc.) of the program, having a global scope.

Reflection

What is the most important thing you learned from this and why?

The most important thing I learned is modular - programming via functions and procedures. It breaks down large problems into smaller, more manageable segments, making code organization, comprehension, and maintenance easier.

How do functions and procedures help you approach the development of larger programs?

Functions and procedures are vital for larger programs as they boost code reusability and readability. Reusability means avoiding redundant code writing, and meaningful names help in quickly understanding code functionality.

How do parameters make your functions and procedures more useful? Give examples from your work.

Parameters enhance the utility of functions and procedures by offering flexibility. They allow functions to work with diverse data values without code rewrite.

Why are global variables considered poor practice, when global constants are considered good practice?

Global variables are deemed poor practice due to potential unexpected side - effects and reduced code modularity. Multiple code parts can modify them, causing hard - to - debug bugs. Global constants, however, are good as they define unchanging

values used throughout the program, improving code comprehensibility and maintainability.

References

Generative AI acknowledgement

No Gen AI be used in this p6 work.

<It's important to acknowledge any use of generative artificial intelligence (gen AI) in your work. If you used gen AI, please state which AI model you used (e.g. ChatGPT) and how you used it in your work.

For more information about how to acknowledge the use of AI, see here:

[>](https://www.monash.edu/student-academic-success/build-digital-capabilities/create-online/acknowledging-the-use-of-generative-artificial-intelligence)

This file has additional line breaks applied by OnTrack because they contain lines longer than the configured limit. Lines over 1000 characters long have been truncated to limit PDF page count. The orginal submission can be retrieved via the "Download Uploaded Files" function.

```
1 #include "splashkit.h"
2 // Using stoi and to_string from std namespace for string/int conversions
3 using std::stoi;
4 using std::to_string;
5
6 // Reads a string input from the user
7 // Parameter: prompt - message to display to the user
8 string read_string(string prompt)
9 {
10     write(prompt);
11     return read_line();
12 }
13
14 // Reads and validates integer input from the user
15 // Parameter: prompt - message to display to the user
16 int read_integer(string prompt)
17 {
18     string line;
19     line = read_string(prompt);
20     // Loop until valid integer input is received
21     while (!is_integer(line))
22     {
23         write("Please enter a integer: ");
24         line = read_string(prompt);
25     }
26     int result = stoi(line);
27     return result;
28 }
29
30 int calculator(int goal, int behinds)
31 {
32     int score;
33     score = goal * 6 + behinds; // Corrected formula (original had +6 which seemed
34     // incorrect)
35     return score;
36 }
37
38 int main()
39 {
40     write_line("Welcome to the AFL score calculator!");
41     write_line("Enter the team 1 details: ");
42
43     string name1 = read_string("name: ");
44     int goal1 = read_integer("goals: ");
45     int behind1 = read_integer("behinds: ");
46     int score1 = calculator(goal1, behind1);
47
48     string name2 = read_string("name: ");
49     int goal2 = read_integer("goals: ");
50     int behind2 = read_integer("behinds: ");
```

```
50     int score2 = calculator(goal2, behind2);
51
52     write_line(name1 + ": " + to_string(goal1) + ", " + to_string(behind1) + ", " +
53     →   to_string(score1));
54     write_line(name2 + ": " + to_string(goal2) + ", " + to_string(behind2) + ", " +
55     →   to_string(score2));
56
57     // Determine and display initial match result
58     if (score1 > score2)
59     {
60         write_line(to_string(score1) + ">" + to_string(score2) + "\nThe " + name1 +
61         →   " wins!");
62     }
63     else if (score1 < score2)
64     {
65         write_line(to_string(score2) + ">" + to_string(score1) + "\nThe " + name2 +
66         →   " wins!");
67     }
68     else
69     {
70         write_line(to_string(score1) + "=" + to_string(score2) + "\nDraw!");
71     }
72
73     string quit;
74     string number;
75     int option;
76
77     // Main menu loop - continues until user chooses to quit
78     while (true)
79     {
80
81         write_line("Menu:");
82         write_line("1: Update " + name1 + " goals");
83         write_line("2: Update " + name1 + " behinds");
84         write_line("3: Update " + name2 + " goals");
85         write_line("4: Update " + name2 + " behinds");
86         write_line("5: Print details");
87         write_line("6: Quit");
88
89         number = read_line();
90         option = stoi(number);
91         while (true)
92         {
93             if (option >= 1 && option <= 6)
94             {
95                 break;
96             }
97             else
98             {
99                 write_line("Please enter number between 1 and 6: ");
100                option = stoi(read_line());
101            }
102        }
103    }
```

```
100     // Process selected menu option
101     if (option == 1)
102     {
103         write_line("goals: ");
104         goal1 = stoi(read_line());
105     }
106     else if (option == 2)
107     {
108         write_line("behinds: ");
109         behind1 = stoi(read_line());
110     }
111     else if (option == 3)
112     {
113         write_line("goals: ");
114         goal2 = stoi(read_line());
115     }
116     else if (option == 4)
117     {
118         write_line("behinds: ");
119         behind2 = stoi(read_line());
120     }
121     else if (option == 5)
122     {
123         int score1 = calculator(goal1, behind1);
124         int score2 = calculator(goal2, behind2);
125         if (score1 > score2)
126         {
127             write_line(to_string(score1) + ">" + to_string(score2) + "\nThe " +
128                         ↳ name1 + " wins!");
129         }
130         else if (score1 < score2)
131         {
132             write_line(to_string(score2) + ">" + to_string(score1) + "\nThe " +
133                         ↳ name2 + " wins!");
134         }
135         else
136         {
137             write_line(to_string(score1) + "=" + to_string(score2) + "\nDraw!");
138         }
139     }
140     else if (option == 6)
141     {
142         write_line("Are you sure you want to quit? [Y/n]:");
143         quit = read_line();
144
145         // Confirm user wants to quit
146         while (true)
147         {
148             if (quit == "Y")
149             {
150                 write_line("Bye!");
151                 return 0;
152             }
153             else if (quit == "n")
```

```
152         {
153             break;
154         }
155     else
156     {
157         write_line("Please enter Y/n");
158         quit = read_line();
159     }
160 }
161 }
162 }
163
164 return 0;
165 }
```

13 Working with Multiples

So far, our programs have been working with variables that store a single value/entity. We need new tools to help us work effectively when we want to work with many values. In this task you will see how we can use the power of the computer to work with multiples by making use of arrays.

Date	Author	Comment
2025/08/06 20:24	Ziyue Meng	Planned date adjusted to 05 Sep.
2025/09/08 13:08	Ziyue Meng	Working On It
2025/09/15 18:46	Ziyue Meng	Ready for Feedback
2025/09/22 14:14	Ganesh Krishnasamy	Can also include what you learned so far in the previous task (P1 - C1) before attempting this task in the prior knowledge and experience.
2025/09/22 14:14	Ganesh Krishnasamy	Fix and Resubmit
2025/09/22 14:17	Ganesh Krishnasamy	Explanation with the issue with this code for finding the max value in an array is incorrect.
2025/09/22 14:18	Ganesh Krishnasamy	Did not explain what is the issue with the code for the sorting problem.
2025/09/25 16:33	Ziyue Meng	Ready for Feedback
2025/09/28 17:56	Ganesh Krishnasamy	image comment
2025/09/28 17:56	Ganesh Krishnasamy	You should create enum that to map to these switch cases in your test your knowledge program.
2025/09/28 17:56	Ganesh Krishnasamy	Fix and Resubmit
2025/09/28 17:58	Ganesh Krishnasamy	image comment
2025/09/28 17:58	Ganesh Krishnasamy	your struct product_data is incorrect.
2025/09/28 17:59	Ganesh Krishnasamy	image comment
2025/09/28 17:59	Ganesh Krishnasamy	your struct for store_data is incorrect.
2025/09/29 13:22	Ziyue Meng	Ready for Feedback
2025/10/01 15:31	Ziyue Meng	I have modified my work and reupload it, thank you for your guidance!
2025/10/01 19:24	Ganesh Krishnasamy	Your hand solution for bubble sort is incorrect. Please re-submit.
2025/10/01 19:24	Ganesh Krishnasamy	Fix and Resubmit
2025/10/01 19:25	Ganesh Krishnasamy	image comment
2025/10/01 19:25	Ganesh Krishnasamy	did not answer this question yet.
2025/10/01 19:25	Ganesh Krishnasamy	Fix and Resubmit
2025/10/02 11:30	Ziyue Meng	This is the answer
2025/10/02 11:30	Ziyue Meng	image comment
2025/10/02 11:44	Ziyue Meng	Ready for Feedback
2025/10/02 11:44	Ziyue Meng	I have modified my work and reupload it, thank you for your guidance!
2025/10/04 21:14	Ganesh Krishnasamy	Fix and Resubmit
2025/10/04 21:14	Ganesh Krishnasamy	There are no comments in your code. Please revise and add documentation to show your intent and help others read it.
2025/10/04 21:14	Ganesh Krishnasamy	image comment
2025/10/04 21:14	Ganesh Krishnasamy	You should use enum that to map to these switch cases in your test your knowledge program.
2025/10/04 21:14	Ganesh Krishnasamy	Fix and Resubmit
2025/10/04 21:15	Ganesh Krish-	image comment

MONASH

INTRODUCTION TO PROGRAMMING

ONTRACK SUBMISSION

Working with Multiples

Submitted By:

Ziyue MENG

zmen0034

2025/10/09 17:13

Tutor:

Ganesh KRISHNASAMY

October 9, 2025



Working with Multiples

Name: Ziyue Meng
Student ID: 36035432
Date: 13/9/2025

Learning Journey and Evidence

Prior Knowledge and Experience

I have a solid foundation in several key aspects of programming. I've mastered the use of pointers and references. Pointers are variables that store memory addresses, which allow for direct memory - manipulation and efficient data - passing between functions. References, on the other hand, act as an alias for an existing variable, providing a more intuitive way to access and modify data in some cases.

I'm also well - versed in loops. For example, the `for` loop is useful when you know exactly how many times you want to iterate. It has a concise syntax where you can initialize a counter, set a condition for continuation, and update the counter in each iteration. The `while` loop is more flexible, as it continues as long as a given condition is true, making it suitable for situations where the number of iterations is not fixed in advance. The `do - while` loop is similar to the `while` loop, but it executes the loop body at least once before checking the condition.

Enumerations (`enum`) are a user - defined data type in programming. They allow you to create a set of named constants. For instance, if you are creating a program about the days of the week, you can define an `enum` like `enum Days {MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, SATURDAY, SUNDAY};` which makes the code more readable and maintainable.

Regarding structures, I can define a structure to encapsulate related data. For example, a "Student" struct containing name, ID, and score. I'm able to properly access its members, which is crucial for handling complex data entities in programming.

However, when it comes to arrays, I still have much to learn. I know absolutely nothing about their representation or application. I'm not clear on how to declare and initialize different types of arrays (such as one - dimensional or two - dimensional arrays), nor do I understand common use cases like iterating through array elements, storing a collection of similar data with arrays, or handling edge cases like array index bounds.

Study to Acquire, Refresh, and/or Extend Capability

To study and master this topic I did the following:

Arrays

Arrays have a fixed size, so we should use a constant to make this available throughout the code

```
#include "splashkit.h"
#include "utilities.h"

using std::to_string;

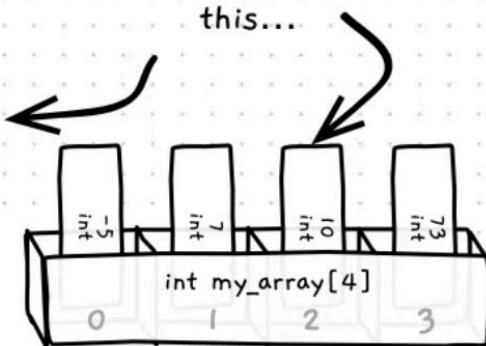
const int SIZE = 4;

int main()
{
    int my_array[SIZE] = {-5, 7, 10, 73};

    my_array[0] = 7;
    my_array[1] = my_array[0] + 3;
}
```

Access each element using its index (starting at 0)

This declares an array of 4 elements (due to SIZE), and initialises them to the provided values. You can picture it like this...



```
for(int i = 0; i < SIZE; i++)
{
    my_array[i] = read_integer("Enter value " + to_string(i+1) + ":");
```

Easily access each element using a for loop!

```
} print_array(my_array, SIZE);
```

Or pass all elements, and the size of the array, to a function or procedure

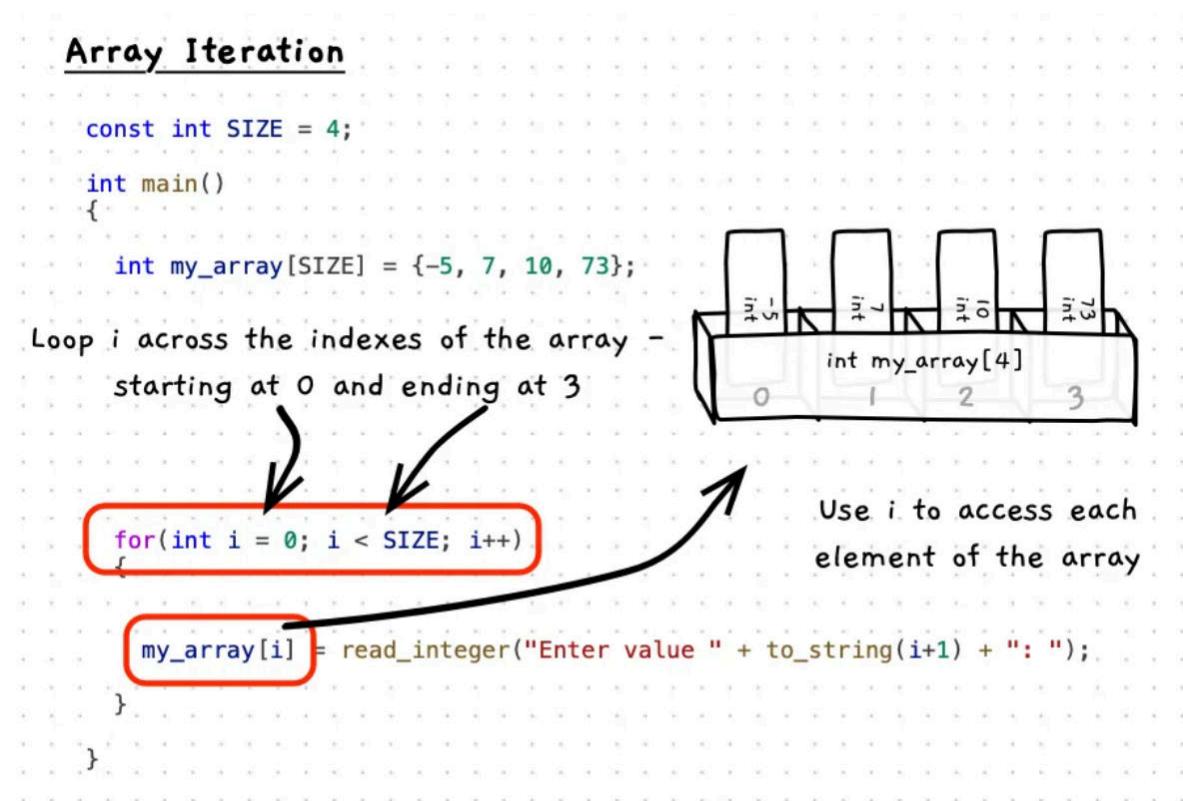
```
27
28 const int SIZE = 4;
29 int main()
30 {
31     int my_array[SIZE] = {-5,7,10,73};
32
33     for(int i = 0; i < SIZE; i++)
34     {
35         my_array[i] = read_integer("Enter value " + to_string(i+1) + ":");
36     }
37
38     return 0;
39 }
```

```

ziyue@XMAS MINGW64 /c/users/ziyue/documents/code1/learnforp8
● $ clang++ hello.cpp utilities.cpp -l splashkit -o hello

ziyue@XMAS MINGW64 /c/users/ziyue/documents/code1/learnforp8
● $ ./hello
Enter value 1:2
Enter value 2:4
Enter value 3:6
Enter value 4:8

```



From the first diagram about "Array Iteration", I learned how to use a `for` - loop to iterate over the indices of an array in order to access and potentially modify each element.

Arrays as parameters

```

void populate(int arr[], int from, int to)
{
    for(int i = from; i <= to; i++)
    {
        arr[i] = read_integer("Enter value " + to_string(i+1) + ": ");
    }
}

int my_array[4]
0   1   2   3
populate(my_array, 1, 3);

```

Arrays are passed as pointers - meaning you can update the passed in array.

You can add `const` if you want to guarantee the array is not changed to the caller.

From the second diagram on "Arrays as parameters", I learned that arrays are passed as pointers in functions, allowing the original array passed in to be updated, and that the `const` keyword can be used to prevent such modifications.

```

void populate(int arr[], int from, int to)
{
    for(int i = from; i <= to; i++)
    {
        arr[i] = read_integer("enter value " + to_string(i+1) + "is: ");
    }
}

```

I have started learning the "In Depth" section.

```
34     bool contains(string names[], int size, string name)
35     {
36         for(int i=0; i<size; i++)
37         {
38             if (to_lowercase(names[i]) == to_lowercase(name))
39             {
40                 write_line("has " + name);
41                 return true;
42             }
43         }
44         write_line("no " + name);
45         return false;
46     }
47
48     string shortest_name(string names[], int size)
49     {
50         string result = names[0];
51         for(int i=1; i<size; i++)
52         {
53             if (length_of(names[i]) < length_of(result))
54             {
55                 result = names[i];
56             }
57         }
58         write_line("Shortest name is " + result);
59         return result;
60     }
61 }
```

This code defines two functions: one to check if a specific string exists in a string array (case-insensitive) and another to find the shortest string in the

array.

```
ziyue@XMAS MINGW64 /c/users/ziyue/documents/code1/learnforp8new
$ clang++ hello.cpp utilities.cpp -l splashkit -o hello

ziyue@XMAS MINGW64 /c/users/ziyue/documents/code1/learnforp8new
$ ./hello
Enter name:ziyue
Enter name:feiyang
Enter name:hangjian
Total length of all name is 20
no Andrew
Shortest name is ziyue

ziyue@XMAS MINGW64 /c/users/ziyue/documents/code1/learnforp8new
$ 
```

```
1 ✓ #include <cstdio>
2   #include "utilities.h"
3
4   const int MAX_NUMBERS = 20;
5
6 ✓ struct number_data
7   {
8     int values[MAX_NUMBERS];
9     int size;
10  };
11
12
13 ✓ void populate(number_data &data)
14  {
15    int size = read_integer("How many numbers do you want to enter: ");
16  ✓ if (size > MAX_NUMBERS)
17  {
18    printf("Sorry, you can only enter %d values. \n", MAX_NUMBERS);
19    size = MAX_NUMBERS;
20  }
21  ✓ else if (size < 0)
22  {
23    size = 0;
24  }
25  data.size = size;
26  ✓ for(int i = 0; i<data.size; i++)
27  {
28    data.values[i] = read_integer("Enter a value: ");
29  }
30 }
31
32
33 ✓ int main()
34  {
35    return 0;
36 }
```

This C++ - code defines a structure `number_data` to hold an array of integers and its size, and a function `populate` that reads user - inputted integers into the array, with validation to ensure the number of inputs doesn't exceed a predefined maximum and is non - negative. The `main` function is currently empty and just returns 0.

```
32 ∵ void print_array(const number_data &data)
33 {
34 ∵     for(int i = 0; i<data.size; i++)
35     {
36         printf("%d: %lf\n", i, data.values[i]);
37     }
38 }
39
40 ∵ int main()
41 {
42     number_data data = {{}, 0};
43
44     populate_array(data);
45
46     print_array(data);
47
48     return 0;
49 }
```

I have started the assignments for the "Test Your Knowledge" section.

```
1 ✓ #include <cstdio>
2   #include "utilities.h"
3
4   const int MAX_PRODUCTS = 100;
5
6   ✓ struct product_data
7   {
8       string names[MAX_PRODUCTS];
9       double costPrice[MAX_PRODUCTS];
10      double sellingPrice[MAX_PRODUCTS];
11      int quantity[MAX_PRODUCTS];
12  };
13
14  ✓ struct store_data
15  {
16      int product_count;
17      double total_sales;
18      double total_profit;
19  };
```

This is C++ code that defines a maximum product limit and two structures: one to store product details (names, prices, quantities) and another to track store metrics (product count, total sales, total profit).

```
64  void populate_array(product_data &products)
65  {
66      int size = read_integer("Enter number of products: ");
67      if (size > MAX_PRODUCTS)
68      {
69          printf("Sorry, you can only enter %d values. \n", MAX_PRODUCTS);
70          size = MAX_PRODUCTS;
71      }
72      else if (size < 0)
73      {
74          size = 0;
75      }
76      for (int i = 0; i < size; i++)
77      {
78          products.names[i] = read_string("Enter product name: ");
79          products.costPrice[i] = read_double("Enter cost price: ");
80          products.sellingPrice[i] = read_double("Enter selling price: ");
81          products.quantity[i] = read_integer("Enter quantity: ");
82      }
83  }
84
```

Create a function to add elements, taking care that each element has a corresponding index and that they are properly aligned.

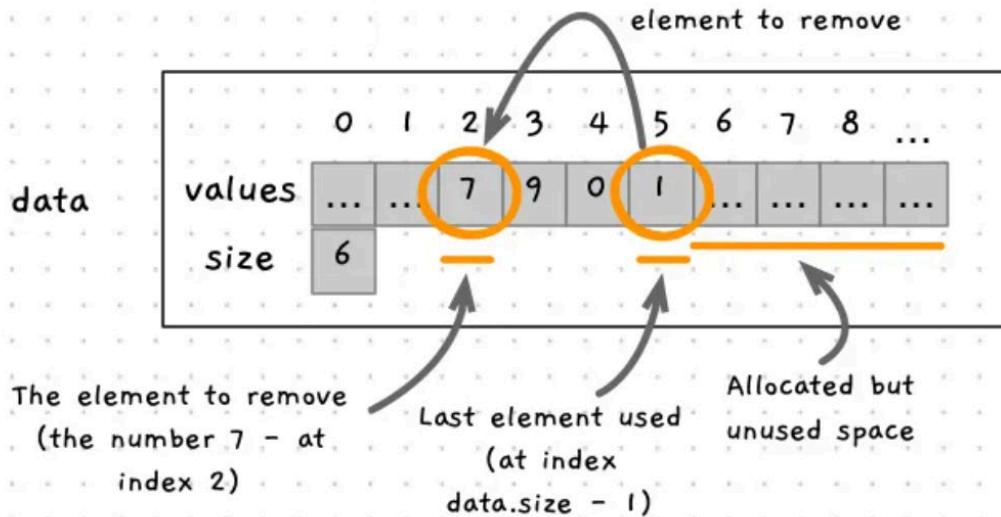
```
85  void print_products(const store_data &store, const product_data &products)
86  {
87      printf("Current Products:\n");
88      for (int i = 0; i < store.product_count; i++)
89      {
90          printf("Index %d: Name: %s, Cost Price: %.2f, Selling Price: %.2f, Quantity: %d\n", i, products.names[i].c_str(), products.costPrice[i], products.sellingPrice[i], products.quantity[i]);
91      }
92 }
```

A function for printing elements has been created.

Option 1 - move last element

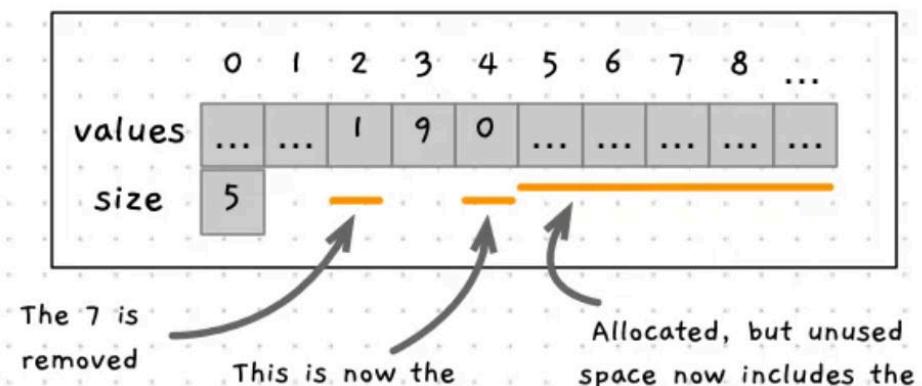
`remove_element(data, 2)`

Copy the last used element over the element to remove



`data.values[index] = data.values[size - 1];`

When finished it will look like this...



```
94 void add_product(store_data &store, product_data &products)
95 {
96     if (store.product_count >= MAX_PRODUCTS)
97     {
98         printf("Cannot add more products. Maximum limit reached.\n");
99         return;
100    }
101    else
102    {
103        products.names[store.product_count] = read_string("Enter product name: ");
104        products.costPrice[store.product_count] = read_double("Enter cost price: ");
105        products.sellingPrice[store.product_count] = read_double("Enter selling price: ");
106        products.quantity[store.product_count] = read_integer("Enter quantity: ");
107        store.product_count++;
108        printf("Product added successfully.\n");
109    }
110}
111
112 void remove_product(store_data &store, product_data &products)
113 {
114     print_products(store, products);
115     printf("please enter an index to remove: ");
116     int index = read_integer("Index:");
117     if (index < 0 || index >= store.product_count)
118     {
119         printf("Invalid index. No product removed.\n");
120         return;
121     }
122     else
123     {
124         products.names[index] = products.names[store.product_count - 1];
125         products.costPrice[index] = products.costPrice[store.product_count - 1];
126         products.sellingPrice[index] = products.sellingPrice[store.product_count - 1];
127         products.quantity[index] = products.quantity[store.product_count - 1];
128         store.product_count--;
129         printf("Product removed successfully.\n");
130     }
131 }
132 }
```

Functions for adding and removing elements have been created, with attention to ensuring that the remove function is implemented in the same way as the method described in the field guide.

```
133 < void print_update()
134 {
135     printf("Menu:\n");
136     printf("1. Update name\n");
137     printf("2. Update cost\n");
138     printf("3. Update the price\n");
139     printf("Update number on hand\n");
140     printf("4. Quit update\n");
141 }
```

The menu of the update part has been created.

```
143 void update_products(store_data &store, product_data &products)
144 {
145     print_products(store, products);
146     int index = read_integer("Enter the index of the product to update: ");
147     if (index < 0 || index >= store.product_count)
148     {
149         printf("Invalid index. No product updated.\n");
150         return;
151     }
152     else
153     {
154         int choice;
155         do
156         {
157             print_update();
158             choice = read_integer("Enter your choice: ");
159             switch(choice)
160             {
161                 case 1:
162                     products.names[index] = read_string("Enter new name: ");
163                     break;
164                 case 2:
165                     products.costPrice[index] = read_double("Enter new cost: ");
166                     break;
167                 case 3:
168                     products.sellingPrice[index] = read_double("Enter new price: ");
169                     break;
170                 case 4:
171                     products.quantity[index] = read_integer("Enter new quantity: ");
172                     break;
173                 case 5:
174                     printf("Exiting update menu.\n");
175                     break;
176                 default:
177                     printf("Invalid choice. Please try again.\n");
178                     break;
179             }
180         }while (choice != 5);
181     }
182 }
```

For the update function, a do-while loop is used here because this process needs to run once first.

```

21  double calculate_sale_value(const store_data &store, const product_data &products)
22  {
23      double sale_value = 0.0;
24      for(int i = 0; i < store.product_count; i++)
25      {
26          sale_value += products.sellingPrice[i] * products.quantity[i];
27      }
28      return sale_value;
29  }
30
31  double calculate_profit(const store_data &store, const product_data &products)
32  {
33      double total_profit = 0.0;
34      for (int i = 0; i < store.product_count; i++)
35      {
36          total_profit += (products.sellingPrice[i] - products.costPrice[i]) * products.quantity[i];
37      }
38      return total_profit;
39  }
40
41  double calculate_store_value(const store_data &store, const product_data &products)
42  {
43      double total_value = 0.0;
44      for (int i = 0; i < store.product_count; i++)
45      {
46          total_value += products.costPrice[i] * products.quantity[i];
47      }
48      return total_value;
49  }
50
51  int count_low_stock(const store_data &store, const product_data &products)
52  {
53      int low_stock_count = 0;
54      for (int i = 0; i < store.product_count; i++)
55      {
56          if (products.quantity[i]<10)
57          {
58              low_stock_count++;
59          }
60      }
61      return low_stock_count;
62  }
63

```

The calculation process has been worked out for the final summary section.

```

184  void print_store_summary(const store_data &store, const product_data &products)
185  {
186      printf("Store Summary:\n");
187      printf("Total Products: %d\n", store.product_count);
188      printf("Total Sales: %.2f\n", calculate_sale_value(store, products));
189      printf("Total Profit: %.2f\n", calculate_profit(store, products));
190      printf("Value of stock on hand: %.2f\n", calculate_store_value(store, products));
191      printf("There are %d products less than 10 in stock.\n", count_low_stock(store, products));
192  }
193
194  void print_menu()
195  {
196      printf("Menu:\n");
197      printf("1. Add a product\n");
198      printf("2. Remove a product\n");
199      printf("3. Update a product\n");
200      printf("4. Print store summary\n");
201      printf("5. Quit\n");
202  }
203

```

```
204 ~ int main()
205 {
206     product_data products = {{}, {}, {}, {}};
207     store_data my_store = {0, 0.0, 0.0};
208     populate_array(products);
209     // Further processing can be done here
210
211     int option;
212     while (option != 5)
213     {
214         print_menu();
215         option = read_integer("Enter your choice: ");
216         switch (option)
217         {
218             case 1:
219                 add_product(my_store, products);
220                 break;
221             case 2:
222                 remove_product(my_store, products);
223                 break;
224             case 3:
225                 update_products(my_store, products);
226                 break;
227             case 4:
228                 print_store_summary(my_store, products);
229                 break;
230             case 5:
231                 printf("Exiting program.\n");
232                 break;
233             default:
234                 printf("Invalid choice. Please try again.\n");
235                 break;
236         }
237     }
238     return 0;
239 }
```

The main function has been created, including a switch part.

```
ziyue@XMAS MINGW64 /c/users/ziyue/documents/code1/p8work
④ $ ./hello
Enter number of products: ziyue
Please enter a whole number.
Enter number of products: 2
Enter product name: ziyue
Enter cost price: 100
Enter selling price: 200
Enter quantity: 2
Enter product name: feiyang
Enter cost price: 200
Enter selling price: 400
Enter quantity: 3
Menu:
1. Add a product
2. Remove a product
3. Update a product
4. Print store summary
5. Quit
Enter your choice: 1
Enter product name: hangjian
Enter cost price: 2
Enter selling price: 4
Enter quantity: 1
Product added successfully.
Enter product name: shiqi
Enter cost price: Please enter a number.
Enter cost price:
```

```
ziyue@XMAS MINGW64 /c/users/ziyue/documents/code1/p8work
○ $ █
```

When running the program, there's an issue: after adding an element, it can't stop and remains in the loop.

```
154     int choice;
155     do
156     {
157         print_update();
158         choice = read_integer("Enter your choice: ");
159         switch(choice)
160         {
161             case 1:
162                 products.names[index] = read_string("Enter new name: ");
163                 break;
164             case 2:
165                 products.costPrice[index] = read_double("Enter new cost: ");
166                 break;
167             case 3:
168                 products.sellingPrice[index] = read_double("Enter new price: ");
169                 break;
170             case 4:
171                 products.quantity[index] = read_integer("Enter new quantity: ");
172                 break;
173             case 5:
174                 printf("Exiting update menu.\n");
175                 break;
176             default:
177                 printf("Invalid choice. Please try again.\n");
178                 break;
179         }
180     }while (choice != 5);
181 }
182 }
```

Then I changed the while loop to do-while loop.

I realized that defining each element in the original structure as an array was quite inconvenient, primarily because this design led to complex data management and maintenance difficulties. When each attribute was stored in separate arrays, the data that should belong to the same product (name, cost price, selling price, and quantity) was scattered across different arrays and had to be synchronized through indices.

This approach presented several significant issues: First, data consistency was hard to maintain - when adding, removing, or moving products, we had to ensure all corresponding indices in every array were updated simultaneously, otherwise data misalignment would occur. Second, code readability suffered - functions required frequent use of lengthy access patterns like `products.names[i]` and `products.costPrice[i]`, making the code verbose and difficult to understand.

By using nested structures, I encapsulated related product attributes within a unified `Product` structure, making each product a complete entity where all attributes naturally remain synchronized.

```
255 // Product data structure
256 struct Product {
257     string name;
258     double costPrice;
259     double sellingPrice;
260     int quantity;
261 };
262
263 // Store data structure containing product array
264 struct StoreData {
265     int product_count;
266     double total_sales;
267     double total_profit;
268     Product products[MAX_PRODUCTS]; // Nested product structure
269 };
```

Additionally, I optimized the switch statement in the main function by using enum to map menu options. This approach significantly improves code readability - using meaningful names like ADD_PRODUCT and REMOVE_PRODUCT instead of magic numbers 1, 2, 3 makes the code's intent clearer. It also enhances maintainability - when new menu options need to be added, we simply add new values to the enum rather than searching and replacing magic numbers throughout the code.

These refactoring efforts not only make the code cleaner and more professional but also greatly improve its reliability and maintainability, demonstrating good software engineering practices.

```
246 // Enum to map main menu options
247 enum MenuOption {
248     ADD_PRODUCT = 1,
249     REMOVE_PRODUCT = 2,
250     UPDATE_PRODUCT = 3,
251     PRINT_SUMMARY = 4,
252     EXIT = 5
253 };
254
```

```
440 // Main function
441 int main() {
442     StoreData my_store = {0, 0.0, 0.0}; // Initialize store data
443     populate_array(my_store); // Initialize product array
444
445     int option;
446     do {
447         print_menu();
448         option = read_integer("Enter your choice: ");
449
450     switch (option) {
451         case ADD_PRODUCT:
452             add_product(my_store);
453             break;
454         case REMOVE_PRODUCT:
455             remove_product(my_store);
456             break;
457         case UPDATE_PRODUCT:
458             update_products(my_store);
459             break;
460         case PRINT_SUMMARY:
461             print_store_summary(my_store);
462             break;
463         case EXIT:
464             printf("Exiting program.\n");
465             break;
466         default:
467             printf("Invalid choice. Please try again.\n");
468             break;
469     }
470 } while (option != EXIT);
471
472 return 0;
473 }
```

```
ziyue@XMAS MINGW64 /c/users/ziyue/documents/code1/p8work
○ $ ./hello
Enter number of products: 2
Enter product name: ziyue
Enter cost price: 100
Enter selling price: 200
Enter quantity: 2
Enter product name: feiyang
Enter cost price: 200
Enter selling price: 400
Enter quantity: 3
Menu:
1. Add a product
2. Remove a product
3. Update a product
4. Print store summary
5. Quit
Enter your choice: 1
Enter product name: hangjian
Enter cost price: 2
Enter selling price: 4
Enter quantity: 3
Product added successfully.
Menu:
1. Add a product
2. Remove a product
3. Update a product
4. Print store summary
5. Quit
Enter your choice: 2
Current Products:
Index 0: Name: hangjian, Cost Price: 2.00, Selling Price: 4.00, Quantity: 3
please enter an index to remove: Index:■
```

The previous issue has been resolved, but now a new problem has emerged: the objects are not being listed completely.

```
64 void populate_array(store_data &store , product_data &products)
65 {
66     store.product_count = read_integer("Enter number of products: ");
67     if (store.product_count > MAX_PRODUCTS)
68     {
69         printf("Sorry, you can only enter %d values. \n", MAX_PRODUCTS);
70         store.product_count = MAX_PRODUCTS;
71     }
72     else if (store.product_count < 0)
73     {
74         store.product_count = 0;
75     }
76     for (int i = 0; i < store.product_count; i++)
77     {
78         products.names[i] = read_string("Enter product name: ");
79         products.costPrice[i] = read_double("Enter cost price: ");
80         products.sellingPrice[i] = read_double("Enter selling price: ");
81         products.quantity[i] = read_integer("Enter quantity: ");
82     }
83 }
```

The correction has been completed. The issue arose because I used "size" to represent the capacity when adding elements, which meant the elements were not being added to the "product_count" (i.e., the count of actual products was not updated). Now the problem has been solved.

```
ziyue@XMAS MINGW64 /c/users/ziyue/documents/code1/p8work
● $ ./hello
Enter number of products: 2
Enter product name: ziyue
Enter cost price: 100
Enter selling price: 200
Enter quantity: 2
Enter product name: feiyang
Enter cost price: 200
Enter selling price: 400
Enter quantity: 3
Menu:
1. Add a product
2. Remove a product
3. Update a product
4. Print store summary
5. Quit
Enter your choice: 1
Enter product name: hangjian
Enter cost price: 2
Enter selling price: 4
Enter quantity: 3
Product added successfully.
Menu:
1. Add a product
2. Remove a product
3. Update a product
4. Print store summary
5. Quit
Enter your choice: 2
Current Products:
Index 0: Name: ziyue, Cost Price: 100.00, Selling Price: 200.00, Quantity: 2
Index 1: Name: feiyang, Cost Price: 200.00, Selling Price: 400.00, Quantity: 3
Index 2: Name: hangjian, Cost Price: 2.00, Selling Price: 4.00, Quantity: 3
please enter an index to remove: Index:2
Product removed successfully.
```

```
1. Add a product
2. Remove a product
3. Update a product
4. Print store summary
5. Quit
Enter your choice: 3
Current Products:
Index 0: Name: ziyue, Cost Price: 100.00, Selling Price: 200.00, Quantity: 2
Index 1: Name: feiyang, Cost Price: 200.00, Selling Price: 400.00, Quantity: 3
Enter the index of the product to update: 1
Update Menu:
1. Update name
2. Update cost
3. Update the price
4. Update number on hand
5. Quit update
Enter your choice: 2
Enter new cost: 300
Update Menu:
1. Update name
2. Update cost
3. Update the price
4. Update number on hand
5. Quit update
Enter your choice: 5
Exiting update menu.
Menu:
1. Add a product
2. Remove a product
3. Update a product
4. Print store summary
5. Quit
Enter your choice: 4
Store Summary:
Total Products: 2
Total Sales: 1600.00
Total Profit: 500.00
Value of stock on hand: 1100.00
There are 2 products less than 10 in stock.
Menu:
1. Add a product
2. Remove a product
3. Update a product
4. Print store summary
5. Quit
Enter your choice: 5
Exiting program.
```

SUCCESSFUL!!!!

Brief Summary of Concepts

Concept	Key Idea / Concept
Array	A collection of elements of the same data type stored in contiguous memory locations.
Array Index	A numerical value used to access individual elements within an array, starting from 0.
Array and For Loop	The for - loop is often used to iterate over the elements of an array, allowing operations on each element.
Multi-dimensional Array	An array that has more than one dimension, typically used to represent tabular or matrix - like data structures.

Hand Execution

Watch the [hand execution video](#) then hand execute the following programs.

The following program code aims to calculate the **maximum** of an array of doubles.

```
double max(const double values[], int size)
{
    double result = 0;

    // Check the rest of the values
    for (int i = 0; i < size; i++)
    {
        if (values[i] > result)
        {
            result = values[i];
        }
    }

    return result;
}
```

Hand execute the above function with the following:

```
double my_data[4] = { 10, 3, 19, 2 };
// Hand execute this call
double value = max ( my_data, 4 );
```

Hand execute the above function with the following:

```
double my_data[4] = { -1, 4, 7, 10 };
// Hand execute this call
double value = max ( my_data, 3 );
```

Hand execute the above function with the following:

```
double my_data[4] = { -1, -4, -10, -5 };
// Hand execute this call
double value = max ( my_data, 4 );
```

① $\text{my_data}[4] = \{ 10, 3, 19, 2 \}$

values

10	3	19	2
----	---	----	---

result

10	10	19	19
----	----	----	----

i

0	1	2	3
---	---	---	---

② $\text{my_data}[4] = \{ -1, 4, 7, 10 \}$

values

-1	4	7	10
----	---	---	----

result

0	4	7	10
---	---	---	----

i

0	1	2	3
---	---	---	---

③ $\text{my_data}[4] = \{ -1, -4, -10, -5 \}$

values

-1	-4	-10	-5
----	----	-----	----

result

0

i

X	X	X	3
---	---	---	---

Value

0

Answer : Value = 0

The following program code aims to **sort** an array of doubles.

```
const int MAX_SIZE = 5;
struct double_array
{
    double data[MAX_SIZE];
    int size;
```

```

};

void sort(double_array &arr)
{
    double temp;
    for (int i = 0; i < arr.size - 1; i++)
    {
        for (int j = 0; j < MAX_SIZE --> arr.size - i - 1; j++)
        {
            if (arr.data[j] > arr.data[j + 1])
            {
                temp = arr.data[j];
                arr.data[j] = arr.data[j + 1];
                arr.data[j + 1] = temp;
            }
        }
    }
}

```

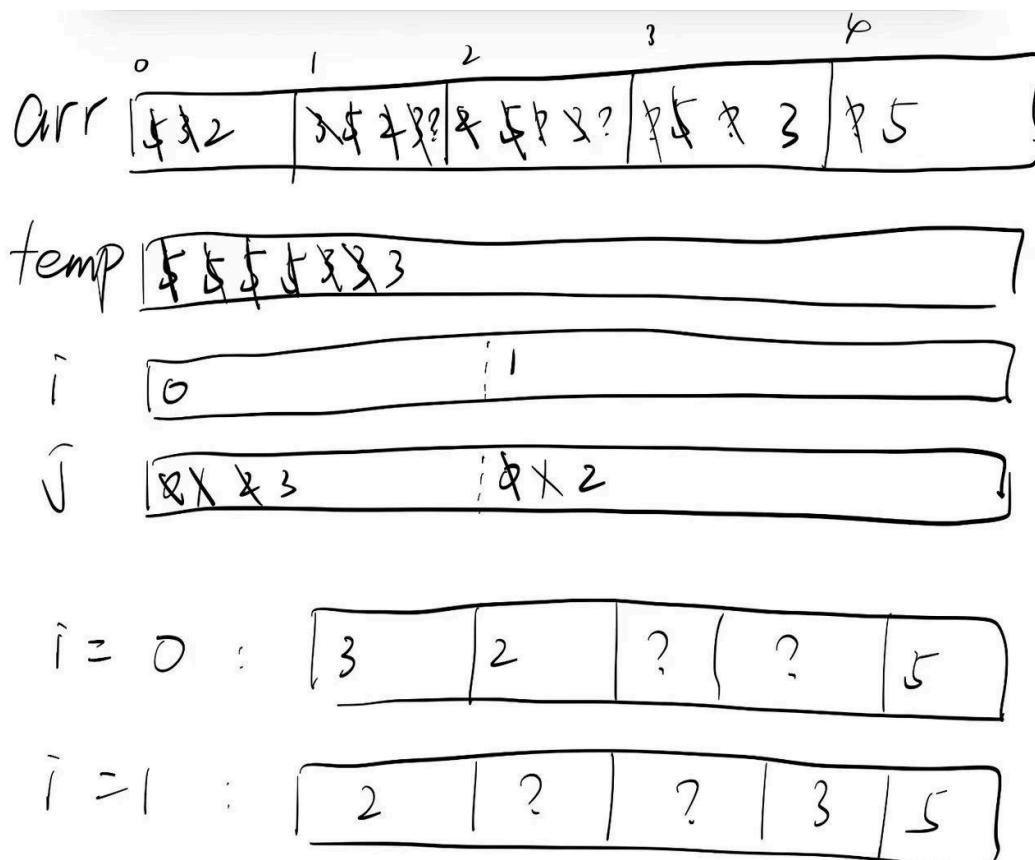
Hand execute the above function with the following:

```

double_array arr = {{5, 3, 2}, 3};

// Hand execute this call
sort(arr);

```



What is the issue with this code?

How can this be fixed?

This program defines a double_array struct and uses bubble sort to arrange its elements in ascending order. It repeatedly compares adjacent values, swapping them if out of order. Each outer loop pass moves the largest remaining value to the end, reducing the range until the array is fully sorted. Just change the upper limit of j to size-1, otherwise it will involve redundant data.

```
utilities.h
main.cpp
```

```
main.cpp > ...
1 const int MAX_SIZE = 5;
2 struct double_array
3 {
4     double data[MAX_SIZE];
5     int size;
6 };
7
8
9 void sort(double_array &arr)
10 {
11     double temp;
12     for (int i = 0; i < arr.size - 1; i++)
13     {
14         for (int j = 0; j < arr.size - i - 1; j++)
15         {
16             if (arr.data[j] > arr.data[j + 1])
17             {
18                 temp = arr.data[j];
19                 arr.data[j] = arr.data[j + 1];
20                 arr.data[j + 1] = temp;
21             }
22         }
23     }
24 }
```

Unknown values

In this hand execution use ? to indicate an unknown value - if you have to compare it with another value assume that ? has the value 0.

Scope of j

Notice that j is declared *within* the “i” for loop. When this goes out of scope, cross out the whole variable and create a new box the next time the declaration is encountered.

Reflection

What is the most important thing you learned from this and why?

The most important thing I learned is the fundamental role of arrays in organizing and managing data. They provide a structured way to store multiple elements of the same type, which is crucial for handling large - scale data in programming.

How do arrays and loops work together? Give some examples of where you have explored this in the programs you have created.

Arrays and loops work together seamlessly. For example, in a program where I needed to sum up all the elements in an array of integers, I used a for - loop. The loop iterated through each index of the array, allowing me to access and add each element to a running total. Another instance was when I had an array of strings representing names, and I wanted to print out each name. The for - loop traversed the array, and with each iteration, it printed the name at the current index.

How do you use for loops with multi-dimensional arrays? How are the number of loops and the number of dimensions connected? What happens when you change the order of the loops?

When using for - loops with multi - dimensional arrays, the number of loops is equal to the number of dimensions. For a two - dimensional array, two nested for - loops are typically used. The outer loop iterates over the rows, and the inner loop iterates over the columns. Changing the order of the loops can significantly change the way the elements are accessed. For instance, if you have a matrix representing a table of data, changing the loop order can change whether you access elements row - by - row or column - by - column.

In C/C++, how does combining the array and its size within a struct help make your code safer and easier to build? Reflect on your experience when combining these.

In C/C++, combining an array and its size within a struct makes the code safer and easier to build. By encapsulating them together, it becomes clear how many elements the array can hold, reducing the risk of buffer - overflows. For example, when I was creating a struct to hold product information as an array, having the size variable within the same struct made it straightforward to perform operations like adding elements, ensuring I didn't exceed the array's capacity. It also made the code more modular and easier to understand and maintain as the relationship between the array and its size was clearly defined.

References

[Generative AI acknowledgement](#)

Copilot

This file has additional line breaks applied by OnTrack because they contain lines longer than the configured limit. Lines over 1000 characters long have been truncated to limit PDF page count. The orginal submission can be retrieved via the "Download Uploaded Files" function.

```
1 #include <cstdio>
2 #include "utilities.h"
3
4 const int MAX_PRODUCTS = 100; // Maximum number of products the store can hold
5
6 /**
7  * Product structure representing a single product in the store inventory
8  * Contains basic product information including pricing and stock details
9  */
10 struct Product
11 {
12     string name;           // Product name/description
13     double costPrice;    // Purchase cost from supplier
14     double sellingPrice; // Retail price for customers
15     int quantity;        // Current stock quantity
16 };
17
18 /**
19  * StoreData structure containing the complete store inventory and financial data
20  * Acts as the main data container for the entire application
21  */
22 struct StoreData
23 {
24     int product_count;      // Current number of products in inventory
25     double total_sales;    // Accumulated total sales (currently unused in
26     // calculations)
27     double total_profit;   // Accumulated total profit (currently unused in
28     // calculations)
29     Product products[MAX_PRODUCTS]; // Array storing all product records
30 };
31
32 /**
33  * Enum for update menu options to improve code readability and maintainability
34  * Maps integer choices to meaningful names
35  */
36 enum UpdateOption
37 {
38     UPDATE_NAME = 1,        // Option to update product name
39     UPDATE_COST = 2,       // Option to update cost price
40     UPDATE_PRICE = 3,      // Option to update selling price
41     UPDATE_QUANTITY = 4,   // Option to update stock quantity
42     QUIT_UPDATE = 5,       // Option to exit update menu
43 };
44
45 /**
46  * Enum for main menu options to improve code readability
47  * Maps menu numbers to meaningful operation names
48  */
49 enum MenuOption
50 {
```

```
49     ADD_PRODUCT = 1,      // Option to add new product
50     REMOVE_PRODUCT = 2,   // Option to remove existing product
51     UPDATE_PRODUCT = 3,   // Option to modify product details
52     PRINT_SUMMARY = 4,    // Option to display store overview
53     EXIT = 5             // Option to terminate application
54 };
55
56 /**
57 * Calculates the total potential sales value of all products in inventory
58 * @return Total sales value as double (selling price × quantity for all products)
59 */
60 double calculate_sale_value(const StoreData &store)
61 {
62     double sale_value = 0.0;
63     // Iterate through all products and accumulate potential sales value
64     for (int i = 0; i < store.product_count; i++)
65     {
66         sale_value += store.products[i].sellingPrice * store.products[i].quantity;
67     }
68     return sale_value;
69 }
70
71 /**
72 * Calculates the total potential profit from all products in inventory
73 * @param store Reference to the StoreData object containing product information
74 * @return Total profit as double ((selling price - cost price) × quantity for all
75 * → products)
76 */
77 double calculate_profit(const StoreData &store)
78 {
79     double total_profit = 0.0;
80     // Calculate profit margin for each product and multiply by quantity
81     for (int i = 0; i < store.product_count; i++)
82     {
83         total_profit += (store.products[i].sellingPrice -
84                         store.products[i].costPrice) * store.products[i].quantity;
85     }
86     return total_profit;
87 }
88
89 /**
90 * Calculates the total inventory value based on cost price
91 * Represents the store's investment in inventory
92 * @param store Reference to the StoreData object containing product information
93 * @return Total store value as double (cost price × quantity for all products)
94 */
95 double calculate_store_value(const StoreData &store)
96 {
97     double total_value = 0.0;
98     // Calculate total cost value of all inventory
99     for (int i = 0; i < store.product_count; i++)
100    {
101        total_value += store.products[i].costPrice * store.products[i].quantity;
102    }
103 }
```

```
101     return total_value;
102 }
103
104 /**
105 * Identifies and counts products with low stock levels (less than 10 units)
106 * Helps in inventory management and reordering decisions
107 * @param store Reference to the StoreData object containing product information
108 * @return Count of products with quantity less than 10
109 */
110 int count_low_stock(const StoreData &store)
111 {
112     int low_stock_count = 0;
113     // Check each product's quantity against low stock threshold
114     for (int i = 0; i < store.product_count; i++)
115     {
116         if (store.products[i].quantity < 10)
117         {
118             low_stock_count++;
119         }
120     }
121     return low_stock_count;
122 }
123
124 /**
125 * Initializes the product array with user input
126 * Sets up the initial inventory for the store
127 * @param store Reference to the StoreData object to be populated
128 */
129 void populate_array(StoreData &store)
130 {
131     // Get the number of products from user with validation
132     int product_count = read_integer("Enter number of products: ");
133
134     // Validate input against maximum capacity
135     if (product_count > MAX_PRODUCTS)
136     {
137         printf("Sorry, you can only enter %d values. \n", MAX_PRODUCTS);
138         store.product_count = MAX_PRODUCTS;
139     }
140     else if (product_count < 0)
141     {
142         store.product_count = 0; // Handle negative input
143     }
144     else
145     {
146         store.product_count = product_count;
147     }
148
149     // Collect detailed information for each product
150     for (int i = 0; i < store.product_count; i++)
151     {
152         store.products[i].name = read_string("Enter product name: ");
153         store.products[i].costPrice = read_double("Enter cost price: ");
154         store.products[i].sellingPrice = read_double("Enter selling price: ");
```

```
155         store.products[i].quantity = read_integer("Enter quantity: ");
156     }
157 }
158
159 /**
160 * Displays all products in the inventory with their complete details
161 * Provides a comprehensive view of current stock
162 * @param store Reference to the StoreData object to display
163 */
164 void print_products(const StoreData &store)
165 {
166     printf("Current Products:\n");
167     // Display each product with index for reference
168     for (int i = 0; i < store.product_count; i++)
169     {
170         printf("Index %d: Name: %s, Cost Price: %.2f, Selling Price: %.2f, Quantity:
171             %d\n",
172             i, store.products[i].name.c_str(), store.products[i].costPrice,
173             store.products[i].sellingPrice, store.products[i].quantity);
174     }
175
176 /**
177 * Adds a new product to the store inventory
178 * Validates available capacity before adding
179 * @param store Reference to the StoreData object to modify
180 */
181 void add_product(StoreData &store)
182 {
183     // Check if store has reached maximum capacity
184     if (store.product_count >= MAX_PRODUCTS)
185     {
186         printf("Cannot add more products. Maximum limit reached.\n");
187         return;
188     }
189
190     // Collect new product information and add to end of array
191     store.products[store.product_count].name = read_string("Enter product name: ");
192     store.products[store.product_count].costPrice = read_double("Enter cost price:
193         ");
194     store.products[store.product_count].sellingPrice = read_double("Enter selling
195         price: ");
196     store.products[store.product_count].quantity = read_integer("Enter quantity: ");
197     store.product_count++; // Increment product count
198     printf("Product added successfully.\n");
199
200 /**
201 * Removes a product from the store inventory
202 * Uses efficient removal by swapping with last element
203 * @param store Reference to the StoreData object to modify
204 */
205 void remove_product(StoreData &store)
206 {
```

```
206     print_products(store); // Show current products for reference
207     printf("Please enter an index to remove: ");
208     int index = read_integer("Index:");
209
210     // Validate index range
211     if (index < 0 || index >= store.product_count)
212     {
213         printf("Invalid index. No product removed.\n");
214         return;
215     }
216
217     // Efficient removal: replace target with last element and reduce count
218     store.products[index] = store.products[store.product_count - 1];
219     store.product_count--;
220     printf("Product removed successfully.\n");
221 }
222
223 /**
224 * Displays the update menu options to the user
225 * Shows available fields that can be modified for a product
226 */
227 void print_update_menu()
228 {
229     printf("Update Menu:\n");
230     printf("%d. Update name\n", UPDATE_NAME);
231     printf("%d. Update cost\n", UPDATE_COST);
232     printf("%d. Update the price\n", UPDATE_PRICE);
233     printf("%d. Update number on hand\n", UPDATE_QUANTITY);
234     printf("%d. Quit update\n", QUIT_UPDATE);
235 }
236
237 /**
238 * Updates product information based on user selection
239 * Provides interactive menu for modifying different product fields
240 * @param store Reference to the StoreData object to modify
241 */
242 void update_products(StoreData &store)
243 {
244     print_products(store); // Display current products
245     int index = read_integer("Enter product index to update: ");
246
247     // Validate product index
248     if (index < 0 || index >= store.product_count)
249     {
250         printf("Invalid index.\n");
251         return;
252     }
253
254     // Interactive update loop
255     int update_option;
256     do
257     {
258         print_update_menu();
259         update_option = read_integer("Enter update option: ");
```

```
260
261     // Use enum for clear, maintainable switch cases
262     switch (update_option)
263     {
264         case UPDATE_NAME:
265             store.products[index].name = read_string("Enter new name: ");
266             break;
267         case UPDATE_COST:
268             store.products[index].costPrice = read_double("Enter new cost price: ");
269             break;
270         case UPDATE_PRICE:
271             store.products[index].sellingPrice = read_double("Enter new selling
272                             price: ");
273             break;
274         case UPDATE_QUANTITY:
275             store.products[index].quantity = read_integer("Enter new quantity: ");
276             break;
277         case QUIT_UPDATE:
278             printf("Quitting update.\n");
279             break;
280         default:
281             printf("Invalid option.\n");
282             break;
283     }
284 } while (update_option != QUIT_UPDATE);
285
286 /**
287 * Displays comprehensive store summary including financial metrics
288 * Provides overview of store performance and inventory status
289 * @param store Reference to the StoreData object to summarize
290 */
291 void print_store_summary(const StoreData &store)
292 {
293     printf("Store Summary:\n");
294     printf("Total Products: %d\n", store.product_count);
295     printf("Total Sales Value: %.2f\n", calculate_sale_value(store));
296     printf("Total Profit: %.2f\n", calculate_profit(store));
297     printf("Total Store Value: %.2f\n", calculate_store_value(store));
298     printf("Low Stock Items: %d\n", count_low_stock(store));
299 }
300
301 /**
302 * Displays the main application menu
303 * Shows all available operations to the user
304 */
305 void print_menu()
306 {
307     printf("\nMain Menu:\n");
308     printf("1. Add Product\n");
309     printf("2. Remove Product\n");
310     printf("3. Update Product\n");
311     printf("4. Print Store Summary\n");
312     printf("5. Exit\n");
```

```
313 }
314
315 /**
316 * Main function - Application entry point
317 * Initializes store data and runs the main program loop
318 */
319 int main()
320 {
321     // Initialize store with empty inventory and zero financials
322     StoreData my_store = {0, 0.0, 0.0};
323     populate_array(my_store); // Load initial product data
324
325     // Main program loop - continues until user chooses to exit
326     int option;
327     do
328     {
329         print_menu();
330         option = read_integer("Enter your choice: ");
331
332         // Process user selection using enum for clarity
333         switch (option)
334         {
335             case ADD_PRODUCT:
336                 add_product(my_store);
337                 break;
338             case REMOVE_PRODUCT:
339                 remove_product(my_store);
340                 break;
341             case UPDATE_PRODUCT:
342                 update_products(my_store);
343                 break;
344             case PRINT_SUMMARY:
345                 print_store_summary(my_store);
346                 break;
347             case EXIT:
348                 printf("Exiting program.\n");
349                 break;
350             default:
351                 printf("Invalid choice. Please try again.\n");
352                 break;
353         }
354     } while (option != EXIT);
355
356     return 0; // Successful program termination
357 }
```

14 Memory Management

So far our memory allocations have had to be set at compile time, limiting us to fixed allocations that cannot change in response to user actions. Now, with pointers and references, we have new potential to indirectly access data. We can use this new capability alongside some memory management functions to give us the capability to dynamically allocate and free memory resources for our programs.

Date	Author	Comment
2025/08/06 20:24	Ziyue Meng	Planned date adjusted to 12 Sep.
2025/09/16 19:59	Ziyue Meng	Working On It
2025/10/02 12:27	Ziyue Meng	Ready for Feedback
2025/10/07 20:38	Ganesh Krish-nasamy	I would like to see your programs work at this week's applied session.
2025/10/07 20:38	Ganesh Krish-nasamy	Discuss
2025/10/09 17:06	Ganesh Krish-nasamy	Pls acknowledge that you use Co-Pilot for this task.
2025/10/09 17:08	Ganesh Krish-nasamy	Fix and Resubmit
2025/10/09 17:11	Ganesh Krish-nasamy	create UX print menu for linked list..etc, print linked_list, add node, insert node at index i, delete note
2025/10/09 17:50	Ziyue Meng	Ready for Feedback
2025/10/09 17:50	Ziyue Meng	I have revised my assignment in accordance with your requirements. Please review it, and thank you for your guidance.
2025/10/13 12:52	Ganesh Krish-nasamy	image comment
2025/10/13 12:52	Ganesh Krish-nasamy	your initial list should be empty
2025/10/13 12:52	Ganesh Krish-nasamy	Fix and Resubmit
2025/10/13 23:12	Ziyue Meng	Ready for Feedback
2025/10/13 23:12	Ganesh Krish-nasamy	Assess in Portfolio
2025/10/14 01:55	Ziyue Meng	Ready for Feedback
2025/10/14 01:55	Ganesh Krish-nasamy	Assess in Portfolio
2025/10/15 20:03	Ziyue Meng	I have revised my assignment in accordance with your requirements. Please review it, and thank you for your guidance.
2025/10/16 00:45	Ganesh Krish-nasamy	Fix and Resubmit
2025/10/16 13:45	Ziyue Meng	Ready for Feedback
2025/10/16 13:45	Ganesh Krish-nasamy	Assess in Portfolio
2025/10/16 13:51	Ziyue Meng	I have revised my assignment in accordance with your requirements. Please review it, and thank you for your guidance.
2025/10/21 15:45	Ganesh Krish-nasamy	Discuss
2025/10/21 15:50	Ganesh Krish-nasamy	Your extension of P8 is not working. Please make sure that the program can be compiled and run.
2025/10/22 16:50	Ganesh Krish-nasamy	discussed. all looks good now
2025/10/22 16:50	Ganesh Krish-nasamy	Complete

MONASH

INTRODUCTION TO PROGRAMMING

ONTRACK SUBMISSION

Memory Management

Submitted By:

Ziyue MENG

zmen0034

2025/10/16 13:45

Tutor:

Ganesh KRISHNASAMY

October 16, 2025



Deep Dive Memory

Name: Ziyue Meng
Student ID: 36035432
Date: 1/10/2025

Learning Journey and Evidence

Prior Knowledge and Experience

I have a solid foundation in several key aspects of programming. I've mastered the use of pointers and references. Pointers are variables that store memory addresses, which allow for direct memory manipulation and efficient data passing between functions. References, on the other hand, act as an alias for an existing variable, providing a more intuitive way to access and modify data in some cases.

I'm also well - versed in loops. For example, the for loop is useful when you know exactly how many times you want to iterate. It has a concise syntax where you can initialize a counter, set a condition for continuation, and update the counter in each iteration. The while loop is more flexible, as it continues as long as a given condition is true, making it suitable for situations where the number of iterations is not fixed in advance. The do - while loop is similar to the while loop, but it executes the loop body at least once before checking the condition.

Enumerations (enum) are a user - defined data type in programming. They allow you to create a set of named constants. For instance, if you are creating a program about the days of the week, you can define an enum like enum Days {MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, SATURDAY, SUNDAY}; which makes the code more readable and maintainable.

Regarding structures, I can define a structure to encapsulate related data. For example, a "Student" struct containing name, ID, and score. I'm able to properly access its members, which is crucial for handling complex data entities in programming.

When it comes to arrays, I have a good grasp of their usage. I know how to declare and initialize both one - dimensional and two - dimensional arrays. For a one - dimensional array, it can be done like int numbers[5] = {1, 2, 3, 4, 5}; and for a two - dimensional array, something like int matrix[2][3] = {{1, 2, 3}, {4, 5, 6}} works. I'm familiar with iterating through array elements using loops, which is essential for processing the data they hold. Arrays are great for storing collections of similar data, such as a list of student scores or a series of temperatures. I also understand the importance of being cautious with array index bounds to avoid accessing elements outside the valid range, which can lead to unexpected program behavior.

Study to Acquire, Refresh, and/or Extend Capability

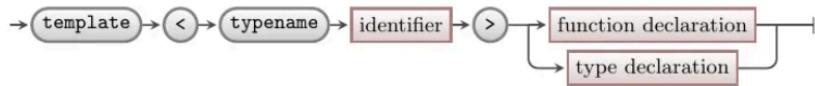
To study and master this topic I did the following:

Dynamic Array: In C/C++

Syntax

C/C++ provides **templates** as their means of implementing generics. You can think of it like adding a placeholder type into your code, and then when you use the generic struct or function, you can indicate what the type is.

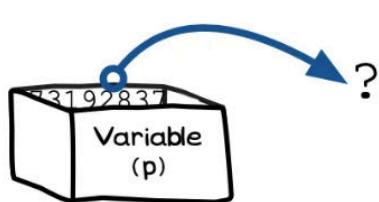
generic decl.



The syntax indicates you want to create a template, where the indicated identifier is now a type name that can be used in the function or struct code. When this function or struct is used, the programmer indicates that the type actually is.

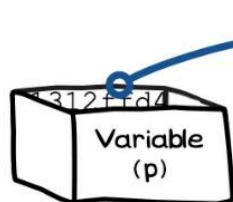
I learned that C++ templates use a specific syntax pattern where you declare `template <typename T>` before a function or struct, creating a placeholder type that gets substituted with actual types when the template is instantiated, enabling

type-safe generic programming.



The Heap

Create p, a Pointer to ...



The Heap

Allocate Space for p

How much space should
be allocated?

When requesting a memory allocation you need to specify the size you want

I learned that heap allocation requires explicitly specifying the memory size needed, and we use pointers to access and manage this dynamically allocated memory,

separating the pointer variable from the actual heap storage location.

In C/C++

C includes two memory allocation functions: `malloc`, and `calloc`. Let's see how each of these work.

Function	Required Arguments	Returns	Description
<code>malloc</code>	the <code>size</code> in bytes that you want.	a pointer (<code>void *</code>)	Allocates memory and returns a pointer to it.
<code>calloc</code>	the number of items, and the size of each item	a pointer (<code>void *</code>)	Allocates and clears memory, returning a pointer to the space allocated.

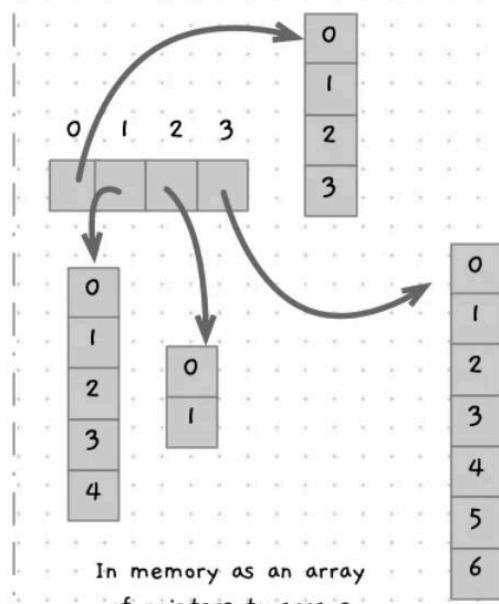
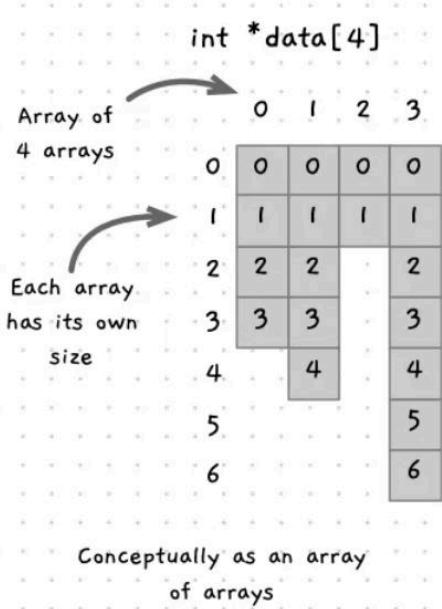
These functions are used in combination with the `sizeof` operator. You can use `sizeof` to get the number of bytes that a data type or variable requires.

Operator: `sizeof`
Returns: (`size_t`) the number of bytes for a type
Parameter: a type or expression

I learned that C provides `malloc` and `calloc` for heap allocation, where `malloc` allocates raw memory of specified bytes while `calloc` allocates and zero-initializes memory for arrays, both requiring `sizeof` to determine proper allocation sizes and

returning void pointers for type casting.

Ragged Array



I learned that ragged arrays are implemented as arrays of pointers where each pointer references a separate dynamically allocated array, allowing each row to have different sizes and creating non-rectangular data structures with flexible dimensions.

In C/C++

Like `malloc` and `calloc`, `realloc` allows you to allocate space from the heap. Using `realloc` you can change (*reallocate*) space on the heap, by passing in the existing pointer and the new size.

Function	Arguments	Returns	Description
<code>realloc</code>	the pointer to reallocate, and new <code>size</code> in bytes.	a pointer (<code>void *</code>) to the new address	Changes the memory allocation and returns a pointer to it.

As with the other memory allocation functions, you need to import the `stdlib.h` header file. The `realloc` function has the following prototype:

```
void *realloc( void *ptr, size_t size )
```



To use this safely you need to think about how it works:

- If it can reallocate at the same location, it will and returns the original pointer.
- If needed to move the allocation, it will do and returns the new address of the data.
- When you are out of memory, or it cannot find a new place to store the data, it will return `nullptr` / `NULL`.

To handle all three of these cases you need to make sure you do not update the original pointer value until you have checked if the `realloc` worked. If you get back `NULL`, then you need to have kept the original pointer so that you can get back to the space on the heap, and potentially free it as you are out of space in memory.

I learned that `realloc` safely resizes heap memory by either expanding the existing block or allocating a new one and copying data, requiring careful handling to preserve the original pointer until success is verified to avoid memory leaks when reallocation fails.

```
11  /*  
12   * @brief A dynamic array struct that contains the size, capacity,  
13   *         and data pointer used to implement this dynamic structure.  
14   *  
15   * @tparam T the type (typename) of data to store in the dynamic array  
16   * @field data a pointer to the data in the dynamic array on the heap  
17   * @field size the number of elements used in the dynamic array  
18   * @field capacity the number of elements the dynamic array can hold  
19   */  
20  
21  template <typename T>  
22  struct dynamic_array__  
23  {  
24      T *data;  
25      unsigned int size;  
26      unsigned int capacity;  
27  };
```

I implemented a generic dynamic array structure using C++ templates.

This taught me how to create type-safe, reusable data structures that separate logical size from physical capacity for efficient memory management.

```

30  /**
31   * @brief Create a new dynamic array with the indicated initial capacity.
32   *        This will return a nullptr if the array cannot be allocated.
33   *
34   * @tparam T the type of data the array will store
35   * @param capacity its initial capacity, with a default value of 50
36   * @return dynamic_array<T>* a pointer to the new dynamic array
37   */
38 template <typename T>
39 dynamic_array__<T> *dynamic_array_create(unsigned int capacity = 50)
40 {
41     // Allocate memory for the dynamic array struct
42     dynamic_array__<T> *arr = (dynamic_array__<T> *)malloc(sizeof(dynamic_array__<T>));
43     if (arr == nullptr)
44     {
45         fprintf(stderr, "Error: could not allocate memory for dynamic array struct.\n");
46         return nullptr;
47     }
48
49     // Allocate memory for the data array
50     arr->data = (T *)malloc(capacity * sizeof(T));
51     if (arr->data == nullptr)
52     {
53         fprintf(stderr, "Error: could not allocate memory for dynamic array data.\n");
54         free(arr);
55         return nullptr;
56     }
57
58     arr->size = 0;
59     arr->capacity = capacity;
60
61     return arr;
62 }
```

I implemented a factory function that creates and initializes dynamic arrays with proper memory allocation and error handling. This taught me the importance of checking both structure and data allocation failures, and ensuring clean resource cleanup when partial allocations fail.

```

64  /**
65   * @brief Free the memory allocated to the dynamic array. Once freed
66   *        the data in the array will no longer be accessible.
67   *
68   * @tparam T the data type of the dynamic array
69   * @param array a pointer to the dynamic array to free
70   */
71 template <typename T>
72 void dynamic_array_free(dynamic_array__<T> *array)
73 {
74     if (array != nullptr)
75     {
76         free(array->data);
77         free(array);
78     }
79 }
```

I created a cleanup function that safely deallocates both the internal data array and the main structure. This taught me proper resource management by checking for null

pointers before freeing and ensuring all allocated memory is released in the correct order.

```
81  /**
82   * @brief Free the memory allocated to the dynamic array. Once freed
83   *        the data in the array will no longer be accessible.
84   *
85   * @tparam T the data type of the dynamic array
86   * @param array a pointer to the dynamic array to free
87   */
88  template <typename T>
89  void delete_dynamic_array(dynamic_array__<T> *&array)
90  {
91      if (array != nullptr)
92      {
93          free(array->data);
94          free(array);
95          array = nullptr;
96      }
97 }
```

I implemented an enhanced deletion function that takes a reference to pointer and sets it to null after freeing. This taught me how to prevent dangling pointers by ensuring the pointer becomes invalid after deallocation, providing safer memory management.

```
99  */
100 * @brief Resize the capacity of the dynamic array.
101 *
102 * If the new capacity is smaller than the current size, the size
103 * will be updated to match the new capacity.
104 *
105 * @tparam T the type of data in the dynamic array
106 * @param array the dynamic array to grow
107 * @param new_capacity the new capacity of the dynamic array
108 * @returns true if this succeeded, or false if it could not reallocate memory
109 */
110 template <typename T>
111 bool dynamic_array_resize(dynamic_array__<T> *array, unsigned int new_capacity)
112 {
113     if (array == nullptr)
114     {
115         return false;
116     }
117
118     T *new_data = (T *)realloc(array->data, new_capacity * sizeof(T));
119     if (new_data == nullptr)
120     {
121         return false;
122     }
123
124     array->data = new_data;
125     array->capacity = new_capacity;
126
127     if (array->size > new_capacity)
128     {
129         array->size = new_capacity;
130     }
131
132     return true;
133 }
134
```

I created a resize function using realloc to efficiently adjust array capacity. This taught me how to handle memory reallocation safely while automatically truncating size when capacity shrinks, ensuring data structure consistency.

```

135  /**
136   * @brief Add an element to the end of the dynamic array
137   *
138   * @tparam T the type of data in the dynamic array
139   * @param value the value to add to the end of the dynamic array
140   * @param array the dynamic array to add the value to
141   * @returns true if this succeeded, false if we cannot resize to fit the new element
142   */
143  template <typename T>
144  bool add(dynamic_array__<T> *array, T value)
145  {
146      if (array == nullptr)
147      {
148          return false;
149      }
150
151      // Resize if necessary
152      if (array->size >= array->capacity)
153      {
154          if (!dynamic_array_resize(array, array->capacity * 2))
155          {
156              return false;
157          }
158      }
159
160      array->data[array->size] = value;
161      array->size++;
162
163      return true;
164  }

```

I implemented an add function that automatically doubles capacity when full.

This taught me the exponential growth strategy for dynamic arrays, balancing memory efficiency with performance by minimizing frequent reallocations.

```

● 166  /**
167   * @brief Resize the capacity of the dynamic array.
168   *
169   * If the new capacity is smaller than the current size, the size
170   * will be updated to match the new capacity.
171   *
172   * @tparam T the type of data in the dynamic array
173   * @param array the dynamic array to grow
174   * @param new_capacity the new capacity of the dynamic array
175   * @returns true if this succeeded, or false if it could not reallocate memory
176   */
177   // Call destructors on all elements removed when reducing size
178
179   // Resize the data in the array
180
181   // Call constructors on all elements added when increasing size
182
183   // Update the size if the new capacity is smaller than the current size
184   */

```

```
1 Click to add a breakpoint name T>
185     bool resize(dynamic_array__<T> *array, unsigned int new_capacity)
186     {
187         if (!array)
188         {
189             return false;
190         }
191
192         // Call destructors on all elements removed when reducing size
193         for(unsigned int i = new_capacity; i < array->size; i++)
194         {
195             array->data[i].~T();
196         }
197         T *new_data = (T *)realloc(array->data, new_capacity * sizeof(T));
198         if (new_data == nullptr)
199         {
200             return false;
201         }
202
203         for(int i = array->size; i < new_capacity; i++)
204         {
205             new(&new_data[i]) T();
206         }
207         array->data = new_data;
208         array->capacity = new_capacity;
209         if (array->size > new_capacity)
210         {
211             array->size = new_capacity;
212         }
213
214         return true;
215     }
216 }
```

I implemented a sophisticated resize function that properly manages object lifecycles. This taught me how to manually call destructors for removed elements and use placement new for new allocations, ensuring proper construction/destruction when working with non-trivial types.

```
218  /**
219  * @brief read and return the value of the indicated element from the dynamic array.
220  *
221  * If the array is not allocated or the index is out of bounds, the function will throw a string exception message.
222  *
223  * @tparam T the type of data in the dynamic array
224  * @param array the dynamic array to remove the element from
225  * @param index the index of the element to remove
226  * @throws a string message if the array is not allocated or the index is invalid
227  */
228 template <typename T>
229 T get(const dynamic_array__<T> *array, unsigned int index)
230 {
231     if (array == nullptr)
232     {
233         throw "Error: dynamic array is not allocated.";
234     }
235
236     if (index >= array->size)
237     {
238         throw "Error: index is out of bounds.";
239     }
240
241     return array->data[index];
242 }
```

I created a safe element accessor that validates both array allocation and index bounds. This taught me how to use exception handling for error reporting in C++ and the importance of const correctness when providing read-only access to data structures.

```
244  /**
245  * @brief set the value of the indicated element from the dynamic array.
246  *
247  * If the index is out of bounds, the function will throw an string exception.
248  *
249  * @tparam T the type of data in the dynamic array
250  * @param array the dynamic array to set the element in
251  * @param index the index of the element to change
252  * @param value the value to set the element to
253  */
254  template <typename T>
255  void set(dynamic_array__<T> *array, unsigned int index, T value)
256  {
257      if (array == nullptr)
258      {
259          throw "Error: dynamic array is not allocated.";
260      }
261
262      if (index >= array->size)
263      {
264          throw "Error: index is out of bounds.";
265      }
266
267      array->data[index] = value;
268  }
269
```

I implemented a safe element modification function that performs the same validation checks as the get function. This taught me the importance of consistent error handling across related functions and maintaining data integrity through proper bounds checking before any write operations.

```
272 int main()
273 {
274     // Create a dynamic array of int
275     // and initialise it to a new dynamic array of 10 elements
276     dynamic_array__<int> *array = dynamic_array_create<int>(10);
277
278     // Add 15 values to the array
279     for(int i = 0; i < 15; i++)
280     {
281         add(array, i);
282     }
283
284
285     // Reprint the size and capacity of the array after adding
286     write_line("size: " + to_string(array->size));
287     write_line("capacity: " + to_string(array->capacity));
288
289     // Print and update the values in the array, using the get and set functions
290     for(unsigned int i = 0; i < array->size; i++)
291     {
292         int value = get(array, i);
293         write_line(to_string(i) + " = " + to_string(value));
294         set(array, i, value * 2);
295     }
296 }
```

```
297     // Attempt to access an element out of bounds using get
298     try
299     {
300         int value = get(array, 99);
301         write_line("ERROR! array[99] = " + to_string(value));
302     }
303     catch (const string &e)
304     {
305         write_line("Caught exception: " + e);
306     }
307
308     // Attempt to access an element out of bounds using set
309     try
310     {
311         set(array, 99, 100);
312         write_line("ERROR: Set array[99] to 100\n");
313     }
314     catch (const string &e)
315     {
316         write_line("Caught exception: " + e);
317     }
318
319     // Free the array and ensure we do not have a dangling pointer
320     delete_dynamic_array(array);
321     array = nullptr;
322
323     return 0;
324 }
```

I successfully implemented and tested a complete template-based dynamic array data structure in C++. The implementation began with creating the core structure containing data pointer, size, and capacity fields, then built all essential operations including creation, destruction, resizing, element access, and modification.

Through the testing phase, I learned how to thoroughly validate a data structure's functionality by demonstrating automatic capacity expansion when adding 15 elements to an initially 10-capacity array, verifying proper element access and modification through iterative get/set operations, and confirming robust error handling by testing out-of-bounds access scenarios that correctly threw and caught exceptions. The tests also showcased proper memory management practices by using the safe deletion function that prevents dangling pointers.

This comprehensive approach taught me the importance of building complete operational workflows - from creation through usage to destruction - and how systematic testing ensures reliability in real-world scenarios while demonstrating the practical benefits of template programming for creating reusable, type-safe data structures.

```
1  #ifndef DYNAMIC_ARRAY_H
2  #define DYNAMIC_ARRAY_H
3
```

```
326  #endif
```

I implemented a header file including guards using preprocessor directives to prevent multiple inclusions.

Test Program:

```
26 // Store data structure containing product array
27 ~ struct StoreData
28 {
29     int product_count;
30     double total_sales;
31     double total_profit;
32     dynamic_array__<Product> *products; // Change to dynamic array pointer
33 };
34
```

I converted the `StoreData`'s fixed product array to a `dynamic_array__<Product>` pointer, shifting from compile-time to runtime memory management. This eliminates size constraints and manual bounds checking, allowing automatic expansion as products are added while delegating memory handling to the dynamic array's built-in allocation system.'

```

86  // Initialize product array
87  void populate_array(StoreData &store)
88  {
89      int product_count = read_integer("Enter number of products: ");
90
91      if (product_count < 0)
92      {
93          store.product_count = 0;
94      }
95      else
96      {
97          store.product_count = product_count;
98      }
99
100     for (int i = 0; i < store.product_count; i++)
101    {
102        Product p;
103        p.name = read_string("Enter product name: ");
104        p.costPrice = read_double("Enter cost price: ");
105        p.sellingPrice = read_double("Enter selling price: ");
106        p.quantity = read_integer("Enter quantity: ");
107        add(store.products, p); // Use add to add product
108    }
109 }
110

```

I replaced direct array assignment with the dynamic array's `add()` function, eliminating the need for manual index management and bounds checking. The code no longer requires pre-allocation of fixed array space since the dynamic array automatically handles memory allocation and expansion as products are added through the loop.

```

2      if (add(store.products, p)) // Use add to add product
3      {
4          store.product_count++;
5          printf("Product added successfully.\n");
6      }

```

I added success checking and explicit `product_count` incrementation after each `add()` operation, since the dynamic array maintains its own internal size separate from the store's tracking. This creates dual tracking where both the dynamic array's size and `store.product_count` need to be synchronized manually after successful additions.

```
197     do
198     {
199         print_update_menu();
200         update_option = read_integer("Enter update option: ");
201
202         Product p = get(store.products, index); // Get current product
203
204         switch (update_option)
205         {
206             case 1:
207                 p.name = read_string("Enter new name: ");
208                 set(store.products, index, p); // Use set to update product
209                 break;
210             case 2:
211                 p.costPrice = read_double("Enter new cost price: ");
212                 set(store.products, index, p);
213                 break;
214             case 3:
215                 p.sellingPrice = read_double("Enter new selling price: ");
216                 set(store.products, index, p);
217                 break;
218             case 4:
219                 p.quantity = read_integer("Enter new quantity: ");
220                 set(store.products, index, p);
221                 break;
222             case 5:
223                 printf("Quitting update.\n");
224                 break;
225             default:
226                 printf("Invalid option.\n");
227                 break;
228         }
229     } while (update_option != 5);
230 }
231 }
```

I replaced direct array element modification with `get()` and `set()` function calls. Instead of directly accessing and updating array elements like `store.products[index].name = new_value`, I now retrieve the entire Product object using `get()`, modify the specific field, then update the array element using `set()`. This adds a layer of safety through the dynamic array's bounds checking but requires

copying the entire product for each field update.'

```
255 ~ int main()
256 {
257     StoreData my_store = {0, 0.0, 0.0, nullptr}; // Initialize store data
258     my_store.products = dynamic_array_create<Product>(); // Create dynamic array
259
260 ~ if (my_store.products == nullptr)
261 {
262     printf("Error: Failed to create dynamic array for products.\n");
263     return 1;
264 }
265
266 populate_array(my_store); // Initialize product array
267 }
```

I added explicit dynamic array creation and null checking in main initialization. Instead of relying on automatic array allocation, I now manually create the products array using `dynamic_array_create<Product>()` and verify its successful creation before proceeding, requiring proper error handling for memory allocation failures.

```
297     // Free dynamic array memory
298     if (my_store.products != nullptr)
299     {
300         delete_dynamic_array(my_store.products);
301     }
302
303     return 0;
304 }
```

I added explicit memory cleanup by calling `delete_dynamic_array()` at program termination. Unlike static arrays that are automatically deallocated, the dynamic array requires manual memory management to free heap-allocated resources and prevent memory leaks.

```
ziyue@XMAS MINGW64 /c/users/ziyue/documents/code1/p8work
● $ ./hello
Enter number of products: 2
Enter product name: ziyue
Enter cost price: 100
Enter selling price: 200
Enter quantity: 2
Enter product name: feiyang
Enter cost price: 200
Enter selling price: 400
Enter quantity: 3
Menu:
1. Add a product
2. Remove a product
3. Update a product
4. Print store summary
5. Quit
Enter your choice: 1
Enter product name: hangjian
Enter cost price: 2
Enter selling price: 4
Enter quantity: 3
Product added successfully.
Menu:
1. Add a product
2. Remove a product
3. Update a product
4. Print store summary
5. Quit
Enter your choice: 2
Current Products:
Index 0: Name: ziyue, Cost Price: 100.00, Selling Price: 200.00, Quantity: 2
Index 1: Name: feiyang, Cost Price: 200.00, Selling Price: 400.00, Quantity: 3
Index 2: Name: hangjian, Cost Price: 2.00, Selling Price: 4.00, Quantity: 3
please enter an index to remove: Index:2
Product removed successfully.
```

```
1. Add a product
2. Remove a product
3. Update a product
4. Print store summary
5. Quit
Enter your choice: 3
Current Products:
Index 0: Name: ziyue, Cost Price: 100.00, Selling Price: 200.00, Quantity: 2
Index 1: Name: feiyang, Cost Price: 200.00, Selling Price: 400.00, Quantity: 3
Enter the index of the product to update: 1
Update Menu:
1. Update name
2. Update cost
3. Update the price
4. Update number on hand
5. Quit update
Enter your choice: 2
Enter new cost: 300
Update Menu:
1. Update name
2. Update cost
3. Update the price
4. Update number on hand
5. Quit update
Enter your choice: 5
Exiting update menu.
Menu:
1. Add a product
2. Remove a product
3. Update a product
4. Print store summary
5. Quit
Enter your choice: 4
Store Summary:
Total Products: 2
Total Sales: 1600.00
Total Profit: 500.00
Value of stock on hand: 1100.00
There are 2 products less than 10 in stock.
Menu:
1. Add a product
2. Remove a product
3. Update a product
4. Print store summary
5. Quit
Enter your choice: 5
Exiting program.
```

SUCCESSFUL ! ! !

Linked List:

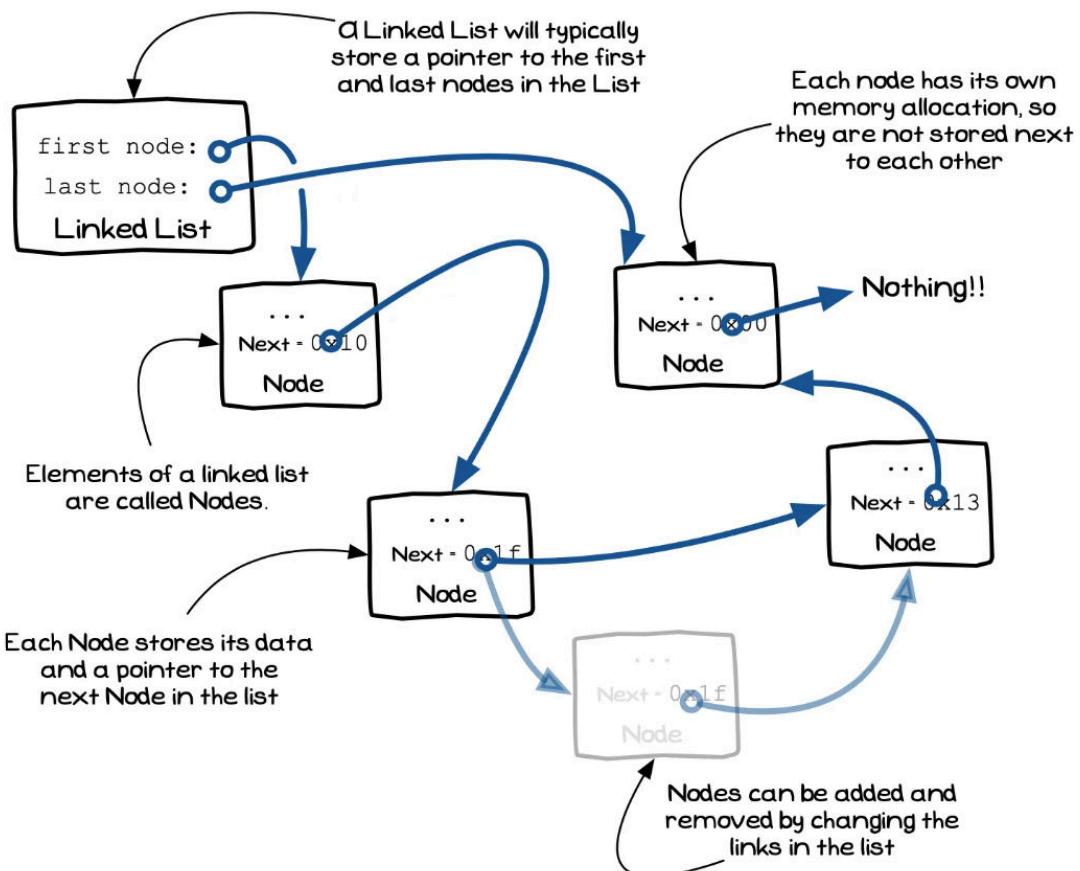


Illustration of a linked list in memory

I learned that linked lists consist of individually allocated nodes containing data and next pointers, with the list structure maintaining first and last node references to enable efficient traversal and modification through pointer manipulation rather than contiguous storage.

```

4 4 /* 
5   * A node is a struct that contains a pointer to the next node,
6   * and a data value.
7   *
8   * @tparam T The type of the data that will be stored in the node.
9   * @field next A pointer to the next node in the list.
10  * @field data The data that is stored in the node.
11  */
12 template <typename T>
13 struct Node
14 {
15     Node<T>* next;
16     T data;
17 };

```

I implemented a template-based Node structure for linked list implementation, containing a data field and next pointer. This follows the same generic programming

approach used in the dynamic array, creating reusable building blocks for chain-based data structures rather than contiguous memory storage.

```
19  /*
20  * A linked list is a struct that contains a pointer to the
21  * first node, and the last node of the list.
22  *
23  * @tparam T The type of the data that will be stored in the list.
24  * @field first A pointer to the first node in the list.
25  * @field last A pointer to the last node in the list.
26  */
27  template <typename T>
28  struct LinkedList
29  {
30      Node<T>* first;
31      Node<T>* last;
32  };
33
```

I created a `LinkedList` structure that maintains pointers to both the first and last nodes, enabling efficient operations at both ends of the list. This two-pointer approach differs from the dynamic array's size/capacity model but serves similar organizational purposes for managing data elements.

```
34  /*
35  * Creates a new linked list.
36  *
37  * @tparam T The type of the data that will be stored in the list.
38  * @return A pointer to the new linked list.
39  */
40  template <typename T>
41  LinkedList<T>* create_linked_list()
42  {
43      // Allocate memory for the linked list
44      LinkedList<T>* list = (LinkedList<T>*)malloc(sizeof(LinkedList<T>));
45
46      // Initialize first and last pointers to null
47      list->first = nullptr;
48      list->last = nullptr;
49      return list;
50 }
```

I implemented a linked list creation function that allocates memory for the list structure and initializes both first and last pointers to null. This follows similar memory allocation patterns as the dynamic array creation but focuses on pointer initialization rather than capacity setup for contiguous storage.

```
52  ~ /**
53   * Add a new node to the end of the list.
54   *
55   * @tparam T The type of the data that will be stored in the list.
56   * @param list A pointer to the linked list to add the node to.
57   * @param data The data to store in the new node.
58   */
59   template <typename T>
60 ~ void add_node(LinkedList<T>* list, T data)
61 {
62     // Allocate memory for new node
63     Node<T>* new_node = (Node<T>*)malloc(sizeof(Node<T>));
64
65     // Initialize new node data and next pointer
66     new_node->data = data;
67     new_node->next = nullptr;
68
69     // If list is empty, set both first and last to new node
70 ~ if (list->first == nullptr)
71 {
72     list->first = new_node;
73     list->last = new_node;
74 }
75 // Otherwise, add to end and update last pointer
76 ~ else
77 {
78     list->last->next = new_node;
79     list->last = new_node;
80 }
81 }
```

I implemented an `add_node` function that creates new nodes and appends them to the linked list end. The function handles both empty list initialization by setting `first/last` pointers and non-empty list updates by modifying the current `last` node's `next` pointer and updating the list's `last` pointer, maintaining efficient O(1) append operations through the tail pointer.

```
83  /**
84   * Delete a linked list and free all memory.
85   *
86   * @tparam T The type of the data that will be stored in the list.
87   * @param list A pointer to the linked list to delete.
88   */
89  template <typename T>
90  void delete_linked_list(LinkedList<T>* list)
91  {
92      Node<T>* current = list->first;
93      Node<T>* next;
94
95      // Traverse the list and free all nodes
96      while (current != nullptr)
97      {
98          next = current->next;
99          free(current);
100         current = next;
101     }
102
103     // Free the list structure itself
104     free(list);
105 }
106
```

I implemented a comprehensive linked list deletion function that traverses the entire list, freeing each node individually before freeing the list structure itself. This demonstrates proper memory cleanup for chain-based structures, contrasting with the dynamic array's single free operation for contiguous memory blocks.

```
107 /**
108  * Print the linked list.
109  *
110  * @tparam T The type of the data that will be stored in the list.
111  * @param list A pointer to the linked list to print.
112  */
113 template <typename T>
114 void print_linked_list(LinkedList<T>* list)
115 {
116     Node<T>* current = list->first;
117
118     // Traverse the list and print each element
119     while (current != nullptr)
120     {
121         // Assuming T can be printed using printf
122         printf("%d ", current->data);
123         current = current->next;
124     }
125
126     printf("\n");
127 }
```

I implemented a print_linked_list function that traverses from the first node through each subsequent node using next pointers, printing each element until reaching the null terminator. This demonstrates sequential access patterns characteristic of linked lists, contrasting with the dynamic array's random access capability through direct indexing.

```
/**  
 * Remove the indicated node from the list.  
 *  
 * @tparam T The type of the data that will be stored in the list.  
 * @param list A pointer to the linked list to remove the node from.  
 * @param node_to_remove The node to remove from the list.  
 */  
template <typename T>  
void remove_node(LinkedList<T>* list, Node<T>* node_to_remove)  
{  
    // Check if list is empty  
    if (list->first == nullptr)  
    {  
        return; // List is empty  
    }  
  
    // Special case: removing the first node  
    if (list->first == node_to_remove)  
    {  
        list->first = list->first->next;  
  
        // If list had only one node, update last pointer  
        if (list->last == node_to_remove)  
        {  
            list->last = nullptr; // List had only one node  
        }  
  
        free(node_to_remove);  
        return;  
    }  
}
```

```

160     // Find the node before the one to remove
161     Node<T>* current = list->first;
162     while (current->next != nullptr && current->next != node_to_remove)
163     {
164         current = current->next;
165     }
166
167     // If node found, remove it
168     if (current->next == node_to_remove)
169     {
170         current->next = node_to_remove->next;
171
172         // Update last pointer if removing the last node
173         if (list->last == node_to_remove)
174         {
175             list->last = current; // Update last if needed
176         }
177
178         free(node_to_remove);
179     }
180 }
```

I implemented a remove_node function that handles three scenarios: empty lists, first node removal, and middle/end node removal. The function maintains list integrity by properly updating first/last pointers when removing boundary nodes and uses sequential traversal to locate the predecessor node for non-first removals, ensuring proper pointer reassignment before freeing the target node memory.

```

182 /**
183 * Main function to test the linked list implementation.
184 */
185 int main()
186 {
187     // Create a new linked list
188     LinkedList<int> *list = create_linked_list<int>();
189
190     // Add some nodes to the list
191     add_node(list, 1);
192     add_node(list, 2);
193     add_node(list, 3);
194     add_node(list, 4);
195
196     // Print the initial list
197     Node<int> *current = list->first;
198     while (current != nullptr)
199     {
200         printf("%d->", current->data);
201         current = current->next;
202     }
203     printf("\n");
204
205     // Remove the third node (value 3)
206     remove_node(list, list->first->next->next);
207 }
```

```
208     // Print the list after removal
209     current = list->first;
210     while (current != nullptr)
211     {
212         printf("%d->", current->data);
213         current = current->next;
214     }
215     printf("\n");
216
217     // Clean up memory
218     delete_linked_list(list);
219
220     return 0;
221 }
```

I created comprehensive linked list testing in main function that demonstrates the complete workflow: creating a list, adding multiple nodes, displaying the list before and after removal operations, and performing proper memory cleanup. The testing includes visual verification of list structure through sequential traversal printing with arrow indicators, showing the actual node linking pattern and confirming correct removal functionality.

```
207  /* 
208   * Print main menu
209   */
210 void print_menu()
211 {
212     printf("\nLinked List Operations:\n");
213     printf("%d. Print Linked List\n", PRINT_LIST);
214     printf("%d. Add Node to End\n", ADD_NODE);
215     printf("%d. Insert Node at Index\n", INSERT_AT_INDEX);
216     printf("%d. Delete Node at Index\n", DELETE_AT_INDEX);
217     printf("%d. Exit\n", EXIT_PROGRAM);
218 }
```

This code defines a menu function for a linked list program. The `print_menu()` function displays a text interface showing all available operations. It lists options for printing the list, adding nodes, inserting at specific positions, deleting nodes, and exiting the program. The menu uses enum constants instead of hard-coded numbers for better code maintenance. This provides the main user interface for interacting with the linked list through console commands.

```
188  /**
189   * Delete entire linked list
190  */
191 template <typename T>
192 void delete_linked_list(LinkedList<T>* list)
193 {
194     Node<T>* current = list->first;
195     Node<T>* next;
196
197     while (current != nullptr)
198     {
199         next = current->next;
200         free(current);
201         current = next;
202     }
203
204     free(list);
205 }
```

This function implements the cleanup routine for a linked list data structure. It systematically traverses through each node in the list, starting from the head pointer, and deallocates all dynamically allocated memory. The function uses a temporary pointer to safely store the reference to the next node before freeing the current node's memory, ensuring no memory leaks occur during the destruction process. After all individual nodes have been properly freed, it finally releases the memory allocated for the main linked list structure itself, completing the comprehensive cleanup of the entire data structure.

```
103     template <typename T>
104     void insert_node_at_index(LinkedList<T>* list, int index, T data)
105     {
106         if (index < 0 || index > list->size)
107         {
108             printf("Error: Index %d out of bounds! List size: %d\n", index, list->size);
109             return;
110         }
111
112         Node<T>* new_node = (Node<T>*)malloc(sizeof(Node<T>));
113         new_node->data = data;
114
115         if (index == 0)
116         {
117             new_node->next = list->first;
118             list->first = new_node;
119             if (list->last == nullptr)
120             {
121                 list->last = new_node;
122             }
123         }
124         else if (index == list->size)
125         {
126             new_node->next = nullptr;
127             list->last->next = new_node;
128             list->last = new_node;
129         }
130         else
131         {
132             Node<T>* current = list->first;
133             for (int i = 0; i < index - 1; i++)
134             {
135                 current = current->next;
136             }
137             new_node->next = current->next;
138             current->next = new_node;
139         }
140         list->size++;
141         printf("Node with value %d inserted at index %d\n", data, index);
142     }
```

This function implements node insertion at a specified position in a linked list. It begins with bounds checking to validate the target index, returning an error if the index is invalid. The code then creates a new node and handles three insertion cases: adding at the beginning by updating the head pointer, appending at the end by modifying the tail reference, or inserting in the middle by traversing to the appropriate position and adjusting node pointers. Finally, it increments the list size and confirms the successful operation.

```
ziyue@XMAS MINGW64 /c/users/ziyue/documents/code1/c2work/linked_list
● $ clang++ linked_list.cpp -l splashkit -o hello

ziyue@XMAS MINGW64 /c/users/ziyue/documents/code1/c2work/linked_list
● $ ./hello

Linked List Operations:
1. Print Linked List
2. Add Node to End
3. Insert Node at Index
4. Delete Node at Index
5. Exit
Enter your choice (1-5): 1
Linked List (3 nodes): 10 -> 20 -> 30 -> NULL

Linked List Operations:
1. Print Linked List
2. Add Node to End
3. Insert Node at Index
4. Delete Node at Index
5. Exit
Enter your choice (1-5): 2
Enter value to add: 40
Node added to end of list

Linked List Operations:
1. Print Linked List
2. Add Node to End
3. Insert Node at Index
4. Delete Node at Index
5. Exit
Enter your choice (1-5): 1
Linked List (4 nodes): 10 -> 20 -> 30 -> 40 -> NULL

Linked List Operations:
1. Print Linked List
2. Add Node to End
3. Insert Node at Index
4. Delete Node at Index
5. Exit
Enter your choice (1-5): 3
Enter index to insert at: 1
Enter value to insert: 15
Node with value 15 inserted at index 1
```

```
Linked List Operations:  
1. Print Linked List  
2. Add Node to End  
3. Insert Node at Index  
4. Delete Node at Index  
5. Exit  
Enter your choice (1-5): 1  
Linked List (5 nodes): 10 -> 15 -> 20 -> 30 -> 40 -> NULL  
  
Linked List Operations:  
1. Print Linked List  
2. Add Node to End  
3. Insert Node at Index  
4. Delete Node at Index  
5. Exit  
Enter your choice (1-5): 4  
Enter index to delete: 2  
Node with value 20 deleted from index 2  
  
Linked List Operations:  
1. Print Linked List  
2. Add Node to End  
3. Insert Node at Index  
4. Delete Node at Index  
5. Exit  
Enter your choice (1-5): 1  
Linked List (4 nodes): 10 -> 15 -> 30 -> 40 -> NULL  
  
Linked List Operations:  
1. Print Linked List  
2. Add Node to End  
3. Insert Node at Index  
4. Delete Node at Index  
5. Exit  
Enter your choice (1-5): 5  
Program exited.
```

```
ziyue@XMAS MINGW64 /c/users/ziyue/documents/code1/c2work/linked_list  
$
```

SUCCESSFUL ! ! ! !

Brief Summary of Concepts

Concept / Commands	Key Idea / Concept
Stack	A region of memory used for storing temporary variables and function call information, following the last - in - first - out principle.
Heap	A region of memory used for dynamic memory allocation, where memory is allocated and deallocated by the programmer.
Malloc / new / new []	<code>malloc</code> and <code>new / new []</code> are used to dynamically allocate memory on the heap for a single object or an array of objects respectively.
Realloc	A function used to resize a previously allocated block of memory on the heap.
Free / delete / delete []	<code>free</code> and <code>delete / delete []</code> are used to deallocate memory that was previously allocated by <code>malloc / new / new []</code> respectively.

Memory allocation

```
#include <cstdlib>
#include <string>

using std::string;

struct person_data
{
    string name;
    int age;
};

int main()
{
    int *arr;
    arr = (int *) malloc(10 * sizeof(int));

    if (arr == NULL)
    {
        return 1;
    }

    int *new_arr;
    new_arr = (int *) realloc(arr, 20 * sizeof(int));
    arr = new_arr;
    free(arr);

    person_data *ptr;

    ptr = new person_data();
    ptr->name = "Jane Doe";
    ptr->age = 30;
    return 0;
}
```

The diagram illustrates the memory allocation process in the provided C code. It uses numbered arrows to show the flow of memory operations:

- Step 1: An arrow points to the first `malloc` call, which allocates memory for 10 integers.
- Step 2: An arrow points to the second `malloc` call, which allocates memory for 20 integers, replacing the previous allocation.
- Step 3: An arrow points to the `realloc` call, which resizes the array to 20 integers.
- Step 4: An arrow points to the `if (arr == NULL)` check, used to handle memory allocation failure.
- Step 5: An arrow points to the `free` call, which frees the memory allocated by the first `malloc`.
- Step 6: An arrow points to the `new` operator, used to create a new `person_data` object.
- Step 7: An arrow points to the assignment of member variables `name` and `age`.
- Step 8: An arrow points to the `return 0;` statement at the end of the function.

	Response
1: Explain how arr and malloc are working here, and how this relates to the heap?	<code>arr</code> is a pointer, and <code>malloc</code> dynamically allocates memory for 10 integers on the heap, with <code>arr</code> storing the address of that allocated memory.
2: Why is the cast needed here?	The cast is needed to convert the <code>void*</code> returned by <code>malloc</code> to <code>int*</code> so <code>arr</code> can properly point to the allocated integer array.
3: What is sizeof and why are we multiplying it here?	<code>sizeof</code> gives the size of a data type; multiplying by 10 ensures enough heap memory is allocated for 10 integers.
4: Why test arr here?	We test <code>arr</code> to check if <code>malloc</code> successfully allocated memory (if <code>arr</code> is <code>NULL</code> , allocation failed).
5: Explain what occurs with realloc here and why new_arr is needed.	<code>realloc</code> resizes the memory block pointed to by <code>arr</code> to hold 20 integers; <code>new_arr</code> is used to store the potentially new address from <code>realloc</code> .
6: Explain what occurs here when free is called.	When <code>free</code> is called, the memory previously allocated for the integer array (pointed to by <code>arr</code>) is deallocated and returned to the heap.
7: Why should we use new rather than malloc here? What does it do?	We use <code>new</code> for a C++ struct as it calls the constructor; <code>new</code> dynamically allocates memory for a <code>person_data</code> object on the heap and initializes it.
8: What do the -> symbols mean here? How would you use this?	The <code>-></code> symbol is used to access members of a struct through a pointer; here, it accesses <code>name</code> and <code>age</code> of the <code>person_data</code> object pointed to by <code>ptr</code> .

Illustrate insert in your linked list

```
test.cpp > main()
1 #include <stdlib.h>
2 #include <stdio.h>
3
4 template <typename T>
5 struct Node
6 {
7     Node<T>* next;
8     T data;
9 };
10
11 template <typename T>
12 struct LinkedList
13 {
14     Node<T>* first;
15     Node<T>* last;
16 };
17
18 template <typename T>
19 LinkedList<T>* create_linked_list()
20 {
21     LinkedList<T>* list = (LinkedList<T>*)malloc(sizeof(LinkedList<T>));
22     list->first = nullptr;
23     list->last = nullptr;
24     return list;
25 }
26
27 template <typename T>
28 void add_node(LinkedList<T>* list, T data)
29 {
30     Node<T>* new_node = (Node<T>*)malloc(sizeof(Node<T>));
31     new_node->data = data;
32     new_node->next = nullptr;
33
34     if (list->first == nullptr)
35     {
36         list->first = new_node;
37         list->last = new_node;
38     }
39     else
40     {
41         list->last->next = new_node;
42         list->last = new_node;
43     }
44 }
```

```
46  template <typename T>
47  void insert_node(LinkedList<T>* list, Node<T>* after_node, T data)
48  {
49      // Create new node
50      Node<T>* new_node = (Node<T>*)malloc(sizeof(Node<T>));
51      new_node->data = data;
52
53      // Case 1: Insert at beginning (when after_node is nullptr or list is empty)
54      if (after_node == nullptr || list->first == nullptr)
55      {
56          new_node->next = list->first;
57          list->first = new_node;
58
59          // If list was empty, update last pointer
60          if (list->last == nullptr)
61          {
62              list->last = new_node;
63          }
64          return;
65      }
66
67      // Case 2: Insert after specified node
68      new_node->next = after_node->next;
69      after_node->next = new_node;
70
71      // Update last pointer if inserting after the last node
72      if (after_node == list->last)
73      {
74          list->last = new_node;
75      }
76  }
77
78  template <typename T>
79  void print_linked_list(LinkedList<T>* list)
80  {
81      Node<T>* current = list->first;
82      while (current != nullptr)
83      {
84          printf("%d -> ", current->data);
85          current = current->next;
86      }
87      printf("NULL\n");
88  }
89
```

```
90 ~ int main()
91 {
92     // Create a new linked list
93     LinkedList<int>* list = create_linked_list<int>();
94
95     printf("Initial list: ");
96     print_linked_list(list);
97
98     // Test 1: Insert into empty list (insert at beginning)
99     printf("\n1. Insert 10 into empty list:\n");
100    insert_node<int>(list, nullptr, 10);
101    print_linked_list(list);
102    // Expected: 10 -> NULL
103
104    // Test 2: Insert at beginning of non-empty list
105    printf("\n2. Insert 5 at beginning:\n");
106    insert_node<int>(list, nullptr, 5);
107    print_linked_list(list);
108    // Expected: 5 -> 10 -> NULL
109
110    // Test 3: Insert in the middle
111    printf("\n3. Insert 7 after first node:\n");
112    insert_node(list, list->first, 7);
113    print_linked_list(list);
114    // Expected: 5 -> 7 -> 10 -> NULL
115
116    // Test 4: Insert at the end
117    printf("\n4. Insert 15 after last node:\n");
118    insert_node(list, list->last, 15);
119    print_linked_list(list);
120    // Expected: 5 -> 7 -> 10 -> 15 -> NULL
121
122    // Test 5: Insert in the middle again
123    printf("\n5. Insert 12 after node with value 10:\n");
124    Node<int>* current = list->first;
125    while (current != nullptr && current->data != 10)
126    {
127        current = current->next;
128    }
129    if (current != nullptr)
130    {
131        insert_node(list, current, 12);
132    }
133    print_linked_list(list);
134    // Expected: 5 -> 7 -> 10 -> 12 -> 15 -> NULL
```

```

135     // Expected: 0
136     return 0;
137 }
```

Initial Linked list setup:

state: empty list

list → first = nullptr

list → last = nullptr

Execution 1: Insert 10 into empty list,

Linked list : [10] → NULL



Execution 2: Insert 5 at beginning

Linked list : [5] → [10] → NULL



Execution 3: Insert 7 after first node

Linked list : [5] → [7] → [10] → NULL



Execution 4: Insert 15 after

[5] → [7] → [10] → [15] → NULL



Final linked list structure: 5 → 7 → 10 → 15 → NULL

This insert function demonstrates the flexibility of linked lists:

- Insert at beginning: Update first pointer while maintaining list connectivity
- Insert in middle: Re-link next pointers of surrounding nodes
- Insert at end: Update last pointer to maintain list integrity

The function successfully handles all edge cases including empty list insertion, beginning insertion, middle insertion, and end insertion while properly maintaining both first and last pointers throughout all operations.

Reflection

What is the most important thing you learned from this and why?

The most important thing I learned is how linked lists maintain dynamic memory efficiency through pointer manipulation. Unlike arrays that require contiguous memory blocks and expensive reallocations, linked lists can grow and shrink dynamically by simply updating node pointers. This demonstrates the fundamental trade-off between memory flexibility and access time, which is crucial for understanding when to choose different data structures based on application requirements.

What challenges and opportunities do you see with dynamic memory management?

The main challenges include memory leaks, dangling pointers, and fragmentation that can crash programs or degrade performance over time. However, dynamic memory management also presents opportunities for creating flexible, memory-efficient applications that adapt to runtime needs. Proper memory management enables developers to build scalable systems that optimize resource usage while preventing the limitations of static allocation.

References

Generative AI acknowledgement

Copilot is used to help me writing the dynamic_array.h file.

This file has additional line breaks applied by OnTrack because they contain lines longer than the configured limit. Lines over 1000 characters long have been truncated to limit PDF page count. The orginal submission can be retrieved via the "Download Uploaded Files" function.

```
1 #ifndef DYNAMIC_ARRAY_H
2 #define DYNAMIC_ARRAY_H
3
4 #include <cstalib>
5 #include<new>
6 #include "splashkit.h"
7
8 using std::to_string;
9
10
11 /**
12 * @brief A dynamic array struct that contains the size, capacity,
13 * and data pointer used to implement this dynamic structure.
14 *
15 * @tparam T the type (typename) of data to store in the dynamic array
16 * @field data a pointer to the data in the dynamic array on the heap
17 * @field size the number of elements used in the dynamic array
18 * @field capacity the number of elements the dynamic array can hold
19 */
20
21 template <typename T>
22 struct dynamic_array__
23 {
24     T *data;
25     unsigned int size;
26     unsigned int capacity;
27 };
28
29
30 /**
31 * @brief Create a new dynamic array with the indicated initial capacity.
32 * This will return a nullptr if the array cannot be allocated.
33 *
34 * @param T the type of data the array will store
35 * @param capacity its initial capacity, with a default value of 50
36 * @return dynamic_array<T>* a pointer to the new dynamic array
37 */
38 template <typename T>
39 dynamic_array__<T> *dynamic_array_create(unsigned int capacity = 50)
40 {
41     // Allocate memory for the dynamic array struct
42     dynamic_array__<T> *arr = (dynamic_array__<T>
43     ↪ *)malloc(sizeof(dynamic_array__<T>));
44     if (arr == nullptr)
45     {
46         fprintf(stderr, "Error: could not allocate memory for dynamic array struct.\n");
47         return nullptr;
48     }
49     // Allocate memory for the data array
```

```
50     arr->data = (T *)malloc(capacity * sizeof(T));
51     if (arr->data == nullptr)
52     {
53         fprintf(stderr, "Error: could not allocate memory for dynamic array data.\n");
54         free(arr);
55         return nullptr;
56     }
57
58     arr->size = 0;
59     arr->capacity = capacity;
60
61     return arr;
62 }
63
64 /**
65 * @brief Free the memory allocated to the dynamic array. Once freed
66 *        the data in the array will no longer be accessible.
67 *
68 * @tparam T the data type of the dynamic array
69 * @param array a pointer to the dynamic array to free
70 */
71 template <typename T>
72 void dynamic_array_free(dynamic_array__<T> *array)
73 {
74     if (array != nullptr)
75     {
76         free(array->data);
77         free(array);
78     }
79 }
80
81 /**
82 * @brief Free the memory allocated to the dynamic array. Once freed
83 *        the data in the array will no longer be accessible.
84 *
85 * @tparam T the data type of the dynamic array
86 * @param array a pointer to the dynamic array to free
87 */
88 template <typename T>
89 void delete_dynamic_array(dynamic_array__<T> *&array)
90 {
91     if (array != nullptr)
92     {
93         free(array->data);
94         free(array);
95         array = nullptr;
96     }
97 }
98
99 /**
100 * @brief Resize the capacity of the dynamic array.
101 *
102 * If the new capacity is smaller than the current size, the size
103 * will be updated to match the new capacity.
```

```
104 *
105 * @tparam T the type of data in the dynamic array
106 * @param array the dynamic array to grow
107 * @param new_capacity the new capacity of the dynamic array
108 * @returns true if this succeeded, or false if it could not reallocate memory
109 */
110 template <typename T>
111 bool dynamic_array_resize(dynamic_array__<T> *array, unsigned int new_capacity)
112 {
113     if (array == nullptr)
114     {
115         return false;
116     }
117
118     T *new_data = (T *)realloc(array->data, new_capacity * sizeof(T));
119     if (new_data == nullptr)
120     {
121         return false;
122     }
123
124     array->data = new_data;
125     array->capacity = new_capacity;
126
127     if (array->size > new_capacity)
128     {
129         array->size = new_capacity;
130     }
131
132     return true;
133 }
134 /**
135 * @brief Add an element to the end of the dynamic array
136 *
137 * @tparam T the type of data in the dynamic array
138 * @param value the value to add to the end of the dynamic array
139 * @param array the dynamic array to add the value to
140 * @returns true if this succeeded, false if we cannot resize to fit the new element
141 */
142 template <typename T>
143 bool add(dynamic_array__<T> *array, T value)
144 {
145     if (array == nullptr)
146     {
147         return false;
148     }
149
150     // Resize if necessary
151     if (array->size >= array->capacity)
152     {
153         if (!dynamic_array_resize(array, array->capacity * 2))
154         {
155             return false;
156         }
157     }

```

```
158     }
159
160     array->data[array->size] = value;
161     array->size++;
162
163     return true;
164 }
165
166 /**
167 * @brief Resize the capacity of the dynamic array.
168 *
169 * If the new capacity is smaller than the current size, the size
170 * will be updated to match the new capacity.
171 *
172 * @tparam T the type of data in the dynamic array
173 * @param array the dynamic array to grow
174 * @param new_capacity the new capacity of the dynamic array
175 * @returns true if this succeeded, or false if it could not reallocate memory
176 */
177 // Call destructors on all elements removed when reducing size
178
179 // Resize the data in the array
180
181 // Call constructors on all elements added when increasing size
182
183 // Update the size if the new capacity is smaller than the current size
184 template <typename T>
185 bool resize(dynamic_array__<T> *array, unsigned int new_capacity)
186 {
187     if (!array)
188     {
189         return false;
190     }
191
192     // Call destructors on all elements removed when reducing size
193     for(unsigned int i = new_capacity; i < array->size; i++)
194     {
195         array->data[i].~T();
196     }
197     T *new_data = (T *)realloc(array->data, new_capacity * sizeof(T));
198     if (new_data == nullptr)
199     {
200         return false;
201     }
202
203     for(int i = array->size; i < new_capacity; i++)
204     {
205         new(&new_data[i]) T();
206     }
207     array->data = new_data;
208     array->capacity = new_capacity;
209     if (array->size > new_capacity)
210     {
211         array->size = new_capacity;
```

```
212     }
213
214     return true;
215 }
216
217 /**
218 * @brief read and return the value of the indicated element from the dynamic array.
219 *
220 * If the array is not allocated or the index is out of bounds, the function will
221 * → throw a string exception message.
222 *
223 * @tparam T the type of data in the dynamic array
224 * @param array the dynamic array to remove the element from
225 * @param index the index of the element to remove
226 * @throws a string message if the array is not allocated or the index is invalid
227 */
228 template <typename T>
229 T get(const dynamic_array__<T> *array, unsigned int index)
230 {
231     if (array == nullptr)
232     {
233         throw "Error: dynamic array is not allocated.";
234     }
235
236     if (index >= array->size)
237     {
238         throw "Error: index is out of bounds.";
239     }
240
241     return array->data[index];
242 }
243
244 /**
245 * @brief set the value of the indicated element from the dynamic array.
246 *
247 * If the index is out of bounds, the function will throw an string exception.
248 *
249 * @tparam T the type of data in the dynamic array
250 * @param array the dynamic array to set the element in
251 * @param index the index of the element to change
252 * @param value the value to set the element to
253 */
254 template <typename T>
255 void set(dynamic_array__<T> *array, unsigned int index, T value)
256 {
257     if (array == nullptr)
258     {
259         throw "Error: dynamic array is not allocated.";
260     }
261
262     if (index >= array->size)
263     {
264         throw "Error: index is out of bounds.";
```

```
265     }
266
267     array->data[index] = value;
268 }
269
270
271
272 int main()
273 {
274     // Create a dynamic array of int
275     // and initialise it to a new dynamic array of 10 elements
276     dynamic_array__<int> *array = dynamic_array_create<int>(10);
277
278     // Add 15 values to the array
279     for(int i = 0; i < 15; i++)
280     {
281         add(array, i);
282     }
283
284
285     // Reprint the size and capacity of the array after adding
286     write_line("size: " + to_string(array->size));
287     write_line("capacity: " + to_string(array->capacity));
288
289     // Print and update the values in the array, using the get and set functions
290     for(unsigned int i = 0; i < array->size; i++)
291     {
292         int value = get(array, i);
293         write_line(to_string(i) + " = " + to_string(value));
294         set(array, i, value * 2);
295     }
296
297     // Attempt to access an element out of bounds using get
298     try
299     {
300         int value = get(array, 99);
301         write_line("ERROR! array[99] = " + to_string(value));
302     }
303     catch (const string &e)
304     {
305         write_line("Caught exception: " + e);
306     }
307
308     // Attempt to access an element out of bounds using set
309     try
310     {
311         set(array, 99, 100);
312         write_line("ERROR: Set array[99] to 100\n");
313     }
314     catch (const string &e)
315     {
316         write_line("Caught exception: " + e);
317     }
318 }
```

```
319     // Free the array and ensure we do not have a dangling pointer
320     delete_dynamic_array(array);
321     array = nullptr;
322
323     return 0;
324 }
325
326 #endif
```

This file has additional line breaks applied by OnTrack because they contain lines longer than the configured limit. Lines over 1000 characters long have been truncated to limit PDF page count. The orginal submission can be retrieved via the "Download Uploaded Files" function.

```
1 #include <csstdio>
2 #include "utilities.h"
3 #include "dynamic_array.hpp" // Add header include
4
5 // Remove MAX_PRODUCTS constant
6
7 // Enum to map main menu options
8 enum MenuOption
9 {
10     ADD_PRODUCT = 1,
11     REMOVE_PRODUCT = 2,
12     UPDATE_PRODUCT = 3,
13     PRINT_SUMMARY = 4,
14     EXIT = 5
15 };
16
17 // Product data structure
18 struct Product
19 {
20     string name;
21     double costPrice;
22     double sellingPrice;
23     int quantity;
24 };
25
26 // Store data structure containing product array
27 struct StoreData
28 {
29     int product_count;
30     double total_sales;
31     double total_profit;
32     dynamic_array__<Product> *products; // Change to dynamic array pointer
33 };
34
35 // Calculate total sales value
36 double calculate_sale_value(const StoreData &store)
37 {
38     double sale_value = 0.0;
39     for (int i = 0; i < store.product_count; i++)
40     {
41         Product p = get(store.products, i);
42         sale_value += p.sellingPrice * p.quantity;
43     }
44     return sale_value;
45 }
46
47 // Calculate total profit
48 double calculate_profit(const StoreData &store)
49 {
50     double total_profit = 0.0;
```

```
51     for (int i = 0; i < store.product_count; i++)
52     {
53         Product p = get(store.products, i);
54         total_profit += (p.sellingPrice - p.costPrice) * p.quantity;
55     }
56     return total_profit;
57 }
58
59 // Calculate total store inventory value
60 double calculate_store_value(const StoreData &store)
61 {
62     double total_value = 0.0;
63     for (int i = 0; i < store.product_count; i++)
64     {
65         Product p = get(store.products, i);
66         total_value += p.costPrice * p.quantity;
67     }
68     return total_value;
69 }
70
71 // Count low stock products (quantity < 10)
72 int count_low_stock(const StoreData &store)
73 {
74     int low_stock_count = 0;
75     for (int i = 0; i < store.product_count; i++)
76     {
77         Product p = get(store.products, i);
78         if (p.quantity < 10)
79         {
80             low_stock_count++;
81         }
82     }
83     return low_stock_count;
84 }
85
86 // Initialize product array
87 void populate_array(StoreData &store)
88 {
89     int product_count = read_integer("Enter number of products: ");
90
91     if (product_count < 0)
92     {
93         store.product_count = 0;
94     }
95     else
96     {
97         store.product_count = product_count;
98     }
99
100    for (int i = 0; i < store.product_count; i++)
101    {
102        Product p;
103        p.name = read_string("Enter product name: ");
104        p.costPrice = read_double("Enter cost price: ");
```